Санкт-Петербургский государственный университет
Факультет прикладной математики – процессов управления

Горяева Мария Андреевна

Магистерская диссертация

Эмуляция радиоэлектронных устройств и их тестирование

Направление 010900

Прикладный математика и физика
Магистерская программа «Математические и информационные технологии»

Руководитель магистерской программы,
доктор физ.-мат. наук,
профессор Овсянников Д.А.

Научный руководитель,
доктор физ.-мат. наук,
профессор Овсянников Д.А.

Санкт-Петербург
2016

Аннотация

В работе рассматриваются вопросы эмуляции радиоэлектронных устройств и их тестирования. Проанализирован теоретический материал по рассматриваемой тематике.

Рассмотрены этапы формирования контрольно-диагностических тестов на примере реального объекта контроля. Приведена оценка эффективности разработанной тестовой программы.

Приведена статистика по результатам формирования контрольно-диагностических тестов для нескольких объектов контроля, а также произведен анализ сокращения временных затрат на эмуляцию радиоэлектронных устройств и формирование тестовых программ при пополнении базы HDL-компонентов.

Abstract

The paper deals with the simulation of electronic devices and their testing. The theoretical material has been analyzed.

There were considered all stages of the formation of diagnostic tests on the example of real object of control. It was produced the estimation of effectiveness developed test program.

It listed the statistics on the results of the formation of the control and diagnostic tests for a number of objects of verification, as well as performed an analysis of the reduction of time spent on emulation of radio-electronic devices and the formation of test programs when funding base HDL-components.

Оглавление

1 Введение

Надежное функционирование современных радиоэлектронных устройств возможно только при соответствующей организации процессов контроля и технической диагностики, включающей в себя процессы тестирования.

Техническая диагностика является частью технического обслуживания и определяется как «область знаний, охватывающая теорию, методы и средства определения технического состояния объектов».

Объект, техническое состояние которого определяется, называют объектом контроля. Техническое диагностирование представляет собой процесс исследования объекта контроля и определение его технического состояния. Характерными примерами результатов технического диагностирования объекта контроля являются заключения вида: объект контроля исправен, неисправен, работоспособен, не работоспособен, в объекте имеется такая-либо неисправность.

Исправное, неисправное, работоспособное и неработоспособное технические состояния определяются следующим образом[4]:

- исправное состояние – состояние объекта контроля, при котором он соответствует всем требованиям нормативно-технической и конструкторской (проектной) документации;
- неисправное состояние – состояние объекта контроля, при котором он не соответствует хотя бы одному из требований нормативно-технической и (или) конструкторской (проектной) документации;
- работоспособное состояние – состояние объекта контроля, при котором значения всех параметров, характеризующих способность выполнять заданные функции, соответствуют требованиям нормативно-технической и (или) конструкторской (проектной) документации.

- неработоспособное состояние – состояние объекта контроля, при котором значение хотя бы одного параметра, характеризующего способность выполнять заданные функции, не соответствует требованиям нормативно-технической и (или) конструкторской (проектной) документации [4].

В процессе производства, эксплуатации и хранения объектов в них могут появляться и накапливаться неисправности, некоторые из них приводят к тому, что объект перестает отвечать предъявляемым к нему техническим требованиям. Перед использованием объекта по назначению необходимо удостовериться в отсутствии неисправностей, которые могут явиться причиной нарушения его штатной работы. Процесс обнаружения неисправности детализируется в зависимости от режима и особенностей использования объекта и, в соответствии с этим, выделяются следующие задачи обнаружения неисправности:

1) контроль исправности, целью которой является разбраковка, позволяющая отделить исправные изделия от неисправных;

2) контроль работоспособности с целью выяснения, будет ли объект выполнять те функции, для реализации которых он предназначен;

3) контроль корректности функционирования с целью обнаружения неисправностей, которые нарушают корректную работу изделия, применяемого по назначению, в данный момент времени.

Если объект неисправен, то для замены или ремонта неисправных компонентов необходимо локализовать место неисправности: поиск неисправности осуществляется путем выполнения диагностического тестирования объекта контроля и дешифрирования его результатов. Диагностическое тестирование в общем случае состоит из отдельных частей, каждая из которых связана с подачей на объект входного воздействия, измерением выходной реакции объекта и сравнением с выходной реакцией исправного объекта [15]. На основе сравнения делается заключение о техническом состоянии объекта.

Дешифрирование результатов диагностического тестирования направлено на определение неисправностей, наличие каждой из которых в объекте не противоречит его штатной работе в процессе выполнения тестирования.

Процесс тестирования в общем случае представляет собой многократную подачу на объект контроля определенных заранее для каждого конкретного изделия воздействий (входных сигналов), многократных измерений, анализа и сравнения ответов на них.

Системы функционального диагностирования обычно обеспечивают контроль объекта в процессе его применения по назначению, тестового – при производстве и ремонте.

Процесс тестирования характеризуется подаваемым на объект тестовым или рабочим (входным) воздействием и снимаемым с объекта контроля ответной реакцией.

Ответные реакции объекта могут сниматься как с основных выходов объекта контроля, то есть с краевых разъемов, необходимых для применения объекта по назначению, так и с дополнительных выходов, организованных специально для диагностирования (диагностические разъемы), расположенных внутри изделия. Краевые и диагностические разъемы называются контролируемыми выходами, измеряемые на них параметры называются контролируемыми или диагностическими параметрами.

Реализация процесса тестирования требует источников тестового воздействия, измерительных устройств (автоматизированное тестовое оборудование) и устройств связи источников воздействий и измерительных устройств с объектом – переходными устройствами. Для управления средствами тестирования и анализа реакции объекта контроля применяют ЭВМ с установленным на ней программным обеспечением, позволяющим производить тестирование объекта контроля [19].

Для тестирования необходимо заранее подготовить некоторые данные – программное обеспечение, содержащее информацию о подаваемых тестовых

последовательностях и ответные реакции исправного объекта контроля. Их качественное и быстрое получение невозможно без использования вычислительной техники и программных средств моделирования (P-CAD, Altium Designer, Altera Quartus II).

Достижение высоких показателей надежности современных радиоэлектронных устройств невозможно без применения методов и средств контрольно-диагностического тестирования, в связи с быстро растущей сложностью объектов контроля активно развиваются методы проектирования схем, обеспечивающие хорошую контролепригодность – свойство объекта, характеризующее его пригодность к проведению тестирования заданными средствами тестирования [6].

Если объект контроля хорошо приспособлен для тестирования (контролепригоден), то существенно упрощается формирование тестовых программ и локализация неисправностей.

Среди всех радиоэлектронных устройств различают аналоговые (непрерывные), цифровые и аналого-цифровые (гибридные) объекты контроля: в аналоговых объектах контроля сигналы характеризуются континуальным множеством значений, в цифровых – логическими уровнями (логический «0» и логическая «1»), в аналого-цифровых – имеются сигналы обоих типов[1,2]. Этапы тестирования и подготовки к ней – формирование тестовой программы, а также средства тестирования зависят от вида радиоэлектронного устройства – объекта контроля.

Под цифровым радиоэлектронным устройством понимается набор электронных компонентов, каждый из которых функционирует по собственному алгоритму, а также связей между ними [13]. Алгоритм функционирования электронного устройства определяется алгоритмами функционирования компонентов, связями между модулями и временными последовательностями входных и выходных сигналов [15].

Для контроля работоспособности и диагностики радиоэлектронных цифровых устройств применяются различные образцы автоматизированного тестового оборудования. Актуальность использования автоматизированного тестового оборудования привела к тому, что наряду с зарубежными наработками и на рынке РФ появились серьезные отечественные разработки [18]. К их числу можно отнести контрольно-диагностическую систему УТК-512.

Применение аппаратно-программного комплекса УТК-512 совместно с разработанной в СПбГУ системой автоматизированного проектирования «SimTest» позволяет осуществлять контроль и производить диагностику сложных цифровых радиоэлектронных устройств, а также существенно сократить время разработки контрольно-диагностического теста и его стоимость [16].

Контрольно-диагностический тест устройства представляет собой совокупность значений, зависящих от времени, входных и выходных сигналов устройства. Эта совокупность представляет собой тестовую таблицу или тестовый векторный набор, описывающий правильное соответствие входных и выходных сигналов на краевых разъемах устройства и является тестом, который может быть загружен в установку тестового контроля [19]. Получить тестовый набор для тестируемого устройства можно либо на реальном заведомо исправном экземпляре устройства, подавая на его входные контакты входные последовательности сигналов и фиксируя его реакции на выходных контактах, сравнивая с заведомо известными, либо путем программного моделирования функционирования устройства [15]. Во втором случае тестовая таблица представляет собой результат эмуляции цифрового радиоэлектронного устройства и формирования тестовой программы для некоторой модели последовательности входных сигналов.

Установка тестового контроля взаимодействует с цифровым радиоэлектронным устройством, выступающим в роли объекта контроля, через краевые разъемы, подавая на соответствующие входные контакты входные воздействия и сравнивая ответную реакцию объекта контроля на эти воздействия со значениями, которые определены заранее и являются результатом разработки

тестовой программы [18]. По итогам сравнения делается вывод о работоспособности объекта контроля.

2 Постановка задачи

Целью магистерской диссертации является тестирование цифровой радиоэлектронной аппаратуры.

В работе были поставлены и решены следующие задачи:

- эмуляция цифрового радиоэлектронного устройства;
- разработка тестовой программы для диагностируемого цифрового радиоэлектронного устройства;
- получение максимальной эффективности разработанной тестовой программы;
- отладка тестовой программы на реальном объекте контроля;
- анализ трудозатрат на эмуляцию радиоэлектронных устройств и формирование тестовых программ.

3 Обзор известных решений задачи

Выполняемая в диссертации работа относится к области диагностики технических систем и может быть использована при формировании эффективных диагностических тестов технических систем (в частности – радиоэлектронной аппаратуры) различной степени сложности, а также создает возможность поэтапного создания качественных диагностических тестов.

Известен способ формирования диагностических тестов, основанный на использовании эталонного образца диагностируемого изделия по книге: Надежность и эффективность в технике. Справочник в 10 т. т.9. Техническая диагностика. Под ред. В.В. Клюева, М.: Машиностроение, 1987, с.177-178. - [1]. Суть этого способа заключается в подаче на входы исправного образца контролируемой аппаратуры наборов входных сигналов, соответствующих реальным сочетаниям входных сигналов для каждой штатной ситуации (штатного набора входных сигналов). Далее фиксируют реакции, возникающие на выходах исправного образца и в характеристических (контрольных) промежуточных точках исправного образца данного вида аппаратуры. Зафиксированные значения сочетаний входных сигналов, значений сочетаний выходных сигналов и значений сочетаний сигналов в промежуточных характеристических (контрольных) точках фиксируют для последующего использования при диагностике состояния и диагностике неисправностей данного типа радиоэлектронной аппаратуры [6].

Достоинством рассмотренного способа является простота формирования контрольно-диагностических тестов, что позволяет в дальнейшем определить состояние контролируемых изделий данного типа радиоэлектронной аппаратуры (по принципу: исправен/неисправен) по совпадению определенных заранее выходных сигналов и выходных сигналов проверяемого образца изделий данного типа - для одних и тех же сочетаний входных (тестовых) сигналов.

Недостатками рассмотренного способа является применение для формирования контрольно-диагностических тестов последовательности входных

сигналов, поступающих на входы диагностируемого изделия, только в процессе его штатной работы, т.е. необходимо наличие исправного образца. Однако такие последовательности, как правило, не обеспечивают как оптимальность процесса контроля, так и требуемые полноту диагностирования и глубину поиска места отказа.

Известен также способ формирования диагностических тестов, основанный на использовании специализированного языка для создания тестов по ГОСТ Р 55692-2013: МОДУЛИ ЭЛЕКТРОННЫЕ. Методы составления и отладки тест-программ. Суть способа заключается в разработке на специализированном языке «Ястек» тестовой программы, описывающей последовательность входных сигналов и соответствующих им выходных сигналов в двоичном коде, обеспечивающих контроль неисправности цифровой радиоэлектронной аппаратуры [3].

Достоинством рассмотренного способа является простота и гибкость формирования диагностических тестов, что позволяет формировать тесты требуемого уровня качества.

Недостатком рассмотренного способа является то, что формирование тестов – это длительная, сложная и кропотливая работа, требующая большого внимания, терпения и усидчивости.

Известен «Способ предварительной оценки качества диагностических тестов» по патенту РФ: RU 2475821 C1 от 20.02.2013г., МПК G06F11/22, G05B23/00, заключающийся в том, что на основе описания внутренних частей диагностируемого изделия формируют эквивалентную поведенческую модель соединений, для полученной поведенческой модели диагностируемого изделия формируют комбинации входных тестовых сигналов, для каждой комбинации входных тестовых сигналов определяют параметры сочетаний выходных сигналов, при этом на входы полученной эталонной модели диагностируемого изделия задают соответствующие сочетания входных сигналов в соот-

ветствующей последовательности, заданной в конкретном оцениваемом диагностическом тесте. Для каждого сочетания задаваемых входных сигналов кроме первого определяют параметры сочетаний сигналов отклика на выходах эталонной модели диагностируемого изделия и в характерных промежуточных (контрольных) точках между эталонными моделями составных частей изделия, и, сравнивая сигналы отклика, полученные для предшествующего сочетания задаваемых входных сигналов, определяют наличие изменения значений сигналов отклика[5]. Вычисляют предварительную эффективность диагностического теста, на основе которой принимают предварительное решение о достаточном качестве оцениваемого диагностического теста, в результате которого оцениваемый диагностический тест отправляют на доработку или принимают в эксплуатацию.

Недостатком является то, что он не предназначен для непосредственного создания (формирования) контрольно-диагностических тестов.

4 Исследование и построение решения задачи

В настоящее время для тестирования цифровых радиоэлектронных устройств широко применяются различные образцы автоматического тестового оборудования. Автоматическое тестовое оборудования взаимодействуют с цифровыми радиоэлектронными устройствами, выступающими в качестве объектов контроля, через краевые разъемы, подавая на входные контакты воздействия *X* и сравнивая ответные реакции *Y* на эти воздействия с заранее определенными при формировании тестовой программы реакциями *Z* [13].

По итогам сравнения ответных реакций объекта контроля *Y* и заранее определенных откликов сформированной программной модели *Z* делается вывод о работоспособности диагностируемого изделия.

Совокупность *{X, Z}* входных воздействий и соответствующих заранее определенных реакций представляет собой контрольно-диагностический тест.

Формирование тестовых программ цифровых радиоэлектронных устройств является сложной научно-технической задачей, требующей больших временных затрат, с привлечением специалистов высокой квалификации.

На современном этапе развития науки и техники наиболее предпочтительными способами создания тестов для цифровых радиоэлектронных устройств являются способы, основанные на программном моделировании. Это связано с тем, что программное моделирование, в отличие от других способов, позволяет получать информацию о значениях цифровых сигналов в любой момент выполнения тестов, а также получать временные диаграммы цифровых сигналов в любой точке объекта контроля без наличия реального объекта контроля, а также осуществлять предварительную оценку качества тестов [13].

При программном моделировании вместо реального объекта контроля используется его программная модель. Программная модель формируется на базе известной структуры объекта контроля, представляющей собой схему связей компонентов изделия и моделей этих компонентов (процессоров, про-

граммируемых логических интегральных схем, микросхем памяти и т.д.). Такой уровень детализации модели объекта контроля связан с тем, что при тестировании радиоэлектронной аппаратуры локализацию неисправностей достаточно осуществлять на уровне локализации корпусов компонентов модулей, так как только они подлежат замене при проведении ремонта и замены неисправных компонентов изделия [2]. В связи с этим для эмуляции достаточно использовать поведенческие модели объекта контроля с уровнем детализации до компонентов цифровых радиоэлектронных устройств. При этом поведенческие модели создаются при помощи языков описания аппаратуры VHDL или Verilog HDL. Эти языки позволяют предельно точно описать логическую структуру и функционирование компонентов радиоэлектронных устройств [7].

Для того, чтобы воспользоваться преимуществами программного моделирования (эмуляции) необходимо, чтобы программная модель была адекватна реальному объекту контроля. Проведенный анализ показал, что адекватность всего объекта контроля в основном определяется адекватностью поведенческих моделей (HDL-моделей) компонентов объекта контроля и связей между ними. Опыт разработки поведенческих моделей различных объектов контроля с целью их использования для создания тестов показал, что создание HDL-моделей компонентов этих объектов контроля имеет свои особенности.

Одной из основных особенностей является использование принципов синтезируемости поведенческих моделей. Это связано с тем, синтезабельные HDL-модели компонентов объектов контроля исключают многие неопределенности, возникающие при переключениях сигналов при использовании несинтезабельных моделей. Типичным примером несинтезабельного описания поведения компонента является присвоение значения одного и того же сигнала в нескольких разных процессах. Синтезабельные модели проще верифицируются - если система синтеза смогла создать структуру соответствующей модели, то по крайней мере она не содержит ошибок в реализации. Однако это

не значит, что разработанная HDL-модель правильно описывает функциональность компонента [2].

Поведенческая модель компонента разрабатывается в функциональном виде, т.е. все ее переменные носят символические функциональные имена. При этом каждая переменная соответствует сигналу реального компонента, а множество сигналов объединяются в функциональные группы. Например, простая микросхема памяти содержит адресную шину, входную шину данных, выходную шину данных и сигналы управления режимами работы (обычно это сигнал выборки и сигнал чтения - записи). Каждая такая шина характеризуется определенной разрядностью. Чтобы разработанная модель описывала конкретную микросхему памяти, у которой каждый сигнал выведен на соответствующую внешний контакт, необходимо привязать сигналы функциональной модели к «ножкам» - контактам микросхемы [13]. Необходимость привязки переменных HDL-моделей компонентов цифровых радиоэлектронных устройств к внешним контактам реального компонента является второй особенностью созданию HDL-моделей компонентов объекта контроля**.**

Иногда в цифровых радиоэлектронных устройствах встречаются схемотехнические решения, реализующие ту или иную цифровую функцию с помощью аналоговых элементов. Типичным примером таких решений является использование транзисторных ключей для согласования уровней сигналов и осуществления логических функций. В HDL-языках нет средств для описания аналоговых элементов, для решения этой проблемы необходимо в схеме электронного цифрового модуля выделить соответствующие схемные решения с использованием аналоговых элементов, определить их функциональность и заменить их на «фиктивные» цифровые компоненты, поддерживающие ту же функциональность. Для этих «фиктивных» компонентов разрабатывается соответствующая поведенческая модель, которая будет использоваться при моделировании устройства. Необходимость реализации функциональности ана-

логовых компонентов объекта контроля поведенческой моделью соответствующего «фиктивного» компонента является еще одной особенностью созданию HDL-моделей компонентов объекта контроля [1].

Следует отметить также, что поскольку при формировании тестов рассматривается только реакция программной модели на цифровые воздействия, то все элементы, связанные с фильтрацией питания (резисторы, конденсаторы, стабилизаторы) необходимо удалить из схемы устройства, т.к. их функциональность не влияет на функциональность объекта контроля в целом.

Отдельно внимания заслуживают вопросы, связанные с использованием моделей различных сверхбольших интегральных схем (программируемых - FPGA, полузаказных -ASIC и т.д.). Разработка функциональности таких интегральных схем, используемая в конкретной схеме, ведется с использованием той или иной системе автоматизированного проектирования, например, Altera Quartus II - для сверхбольших интегральных схем фирмы Altera, Xlinx ISE - для интегральных схем фирмы Xlinx и т.д. [10] При этом для каждого конкретного применения конкретной интегральной схемы формируется поведенческая модель, требуемая для формирования программной модели этой схемы. В результате при формировании тестов для цифровых радиоэлектронных устройств появляется возможность использования готовых и отлаженных на реальных объектах HDL-моделей интегральных схем. При использовании уже готовых HDL-моделей интегральных схем требуется их корректировка и преобразование к тому стандарту языка HDL, который используется при разработке поведенческой модели (HDL-модели) всего объекта контроля. Скорректированные и преобразованные модели требуют дополнительной отладки и верификации. Использование готовых HDL-моделей интегральных схем является еще одной из особенностей создания HDL-моделей компонентов объекта контроля.

Продолжая речь об интегральных схемах, необходимо отметить еще одну особенность создания HDL-моделей компонентов объектов контроля: не-

редко для формирования тестов разработчикам предоставляется цифровое радиоэлектронное устройство без «прошивок» программируемых интегральных схем, что делает практически невозможным создание адекватно смоделированной функциональности, заложенной в эти интегральные схемы в конкретном объекте контроля. В таком случае HDL-модели для каждой такой интегральной схемы необходимо создавать заново. Для этого, исходя из анализа схемы и технической документации, определяется назначение исследуемой микросхемы, ее роль и место в рассматриваемом объекте контроля. На основании полученной информации создается и отлаживается HDL-модель этой интегральной схемы для конкретной ситуации. Эта собственная модель служит основой для разработки собственной временной «прошивки» программируемой интегральной схемы, которая будет использоваться при тестировании. Данная операция становится возможной по причине того, что при тестировании устройства проверяется не его функциональность, а исправность компонентов и связей между ними.

Адекватность HDL-моделей реальным объектам контроля является одной из важнейших задач, решаемых при создании поведенческих моделей. Не является исключением и разработка HDL-моделей компонентов объекта контроля для целей тестирования цифрового радиоэлектронного устройства при помощи автоматизированного тестового оборудования. Использование автоматизированного тестового оборудования значительно облегчает проверку адекватности HDL-моделей объекта контроля реальным изделиям. Суть процесса заключается в «прогоне» разработанных на HDL-моделях тестов на реальных объектах контроля при помощи тестового оборудования. Корректное прохождение теста, т.е. соответствие реальных выходных реакций объекта контроля полученным в результате моделирования реакциям в ответ на подаваемые входные воздействия, также соответствующие модельным, позволяют утверждать, что модель адекватно описывает устройство, по крайней мере, для используемых входных воздействий.

Исходя из вышесказанного можно сделать вывод о том, что при разработке HDL-моделей компонентов объекта контроля с целью их использования для создания тестовых программ имеет свои особенности. К ним относятся использование синтезабельных поведенческих моделей, привязка переменных HDL-моделей к контактам модуля, реализация функциональности аналоговых компонентов объекта контроля, использование готовых HDL-моделей, разработка временных тестовых «прошивок» программируемых интегральных схем и способ подтверждения адекватности поведенческих моделей объекта контроля [1]. Учет этих особенностей позволяет обеспечить адекватность HDL-моделей объектов контроля реальным и как следствие повысить эффективность разрабатываемых тестов и глубину поиска неисправностей.

После формирования поведенческой модели объекта контроля – моделей его компонентов и связей между ними – следует процедура формирования контрольно-диагностического теста.

Рассматриваемый в работе способ формирования диагностических тестов основан на формировании комбинаций входных тестовых сигналов с заданными сочетаниями параметров сигналов и с заданными последовательностями подачи входных сигналов, а также на определении параметров сочетаний выходных сигналов для каждой комбинации входных тестовых сигналов[12]. При этом на основе описания внутренних частей данного типа изделий формируют эквивалентную программную модель соединений, в разрывы эквивалентной программной модели соединений включают предварительно подготовленные поведенческие модели составных частей данного типа изделий. На входы полученной программной модели диагностируемых изделий задают соответствующие сочетания входных сигналов. Для каждого сочетания задаваемых входных сигналов определяют параметры сочетаний сигналов отклика на выходах программной модели диагностируемого изделия и в характерных промежуточных точках между программными моделями составных частей изделия, заносят с помощью ЭВМ значения параметров сигналов отклика, полученные с выходов эталонной модели и с промежуточных точек

программной модели диагностируемого изделия вместе с параметрами соответствующих тестовых входных сигналов в базу данных [14]. Сформированную совокупность входных тестовых сигналов и связанных с ними критериальных эквивалентных выходных сигналов используют для диагностики состояния реальных изделий и диагностики неисправностей составных частей изделий, при этом:

1. Производят анализ схемного решения диагностируемого изделия, по результатам которого осуществляют группировку множества входов на подмножества по признаку - каждому подмножеству подключен один типовой функциональный узел (счетчик, шина, память, тактовый импульс, сигнал сброса и т.д.), при этом каждый вход диагностируемого изделия должен быть привязан к одному из логических интерфейсов.

2. Каждому найденному функциональному узлу ставят в соответствие один из имеющихся в базе данных программных модулей, называемых логическими интерфейсами. Для каждого типового функционального узла логический интерфейс создают и записывают в базу данных заранее.

3. Разрабатывают скрипт, задающий правила формирования входных сигналов. Скрипт представляет собой последовательность операций логических интерфейсов, привязанных к объекту контроля, для которого разрабатывается диагностический тест.

4. Путем обработки скрипта на ЭВМ формируется последовательность сочетаний входных сигналов, которую подают на входы эталонной модели объекта контроля.

5. Для каждого сочетания задаваемых входных сигналов определяются параметры сочетаний сигналов отклика на выходах и в характерных промежуточных точках программной модели объекта контроля.

6. Последовательность сочетаний входных сигналов и соответствующие им реакции на выходах и в характерных промежуточных точках программной модели диагностируемого изделия представляет собой диагностический тест, который с помощью ЭВМ заносят в базу данных.

7. Для полученной версии контрольно-диагностического теста определяют значение его предварительной эффективности. В случае если качество теста не соответствует требуемому уровню, то тест отправляют на доработку. Данный процесс продолжают до тех пор, пока не будет получен тест требуемого качества.

В рассматриваемом способе формирования контрольно-диагностических тестов все входы X диагностируемого изделия

$$X = \left(x_1, x_2, \dots, x_j, \dots, x_p\right) \text{ для } j = 1, \dots, p, \quad (1)$$

представлены в следующем виде

$$X = (X_1, X_2, \dots, X_i, \dots, X_n) \text{ для } i = 1, \dots, n, \quad (2)$$

$$X_i => L_i, \quad (3)$$

$$L_i \in L, \quad (4)$$

$$L = (L_1, L_2, \dots, L_i, \dots, L_Z), \quad (5)$$

$$L_i = \left(l_{i,1}, \dots, l_{i,h}, \dots, l_{i,B_i}\right) \text{ для } h = 1, \dots, B_i, \quad (6)$$

$$x_k => l_{i,h}, \quad (7)$$

$$x_k \in X_i, \quad (8)$$

где p – количество входов диагностируемого изделия;

x_k – k-й вход диагностируемого изделия;

n – количество подмножеств входов диагностируемого изделия;

X_i – i-е подмножество входов X ;

L_i – i-ый логический интерфейс диагностируемого изделия;

L – множество ЛИ диагностируемого изделия;

B_i – количество входов i-го логического интерфейса..;

$l_{i,h}$ – h-й вход i-го L_i,.

Деление множества входов X на подмножества X_i осуществляется путем анализа схемного решения диагностируемого изделия на предмет выделения в нем типовых функциональных узлов, называемых логическими интерфей-

сами, которые связаны со входами диагностируемого изделия. При этом происходит привязка каждого из логических интерфейсов L_i к соответствующему подмножеству X_i (выражение (3)).

Структура представления диагностируемого изделия с использованием логических интерфейсов приведена на рис. 1.

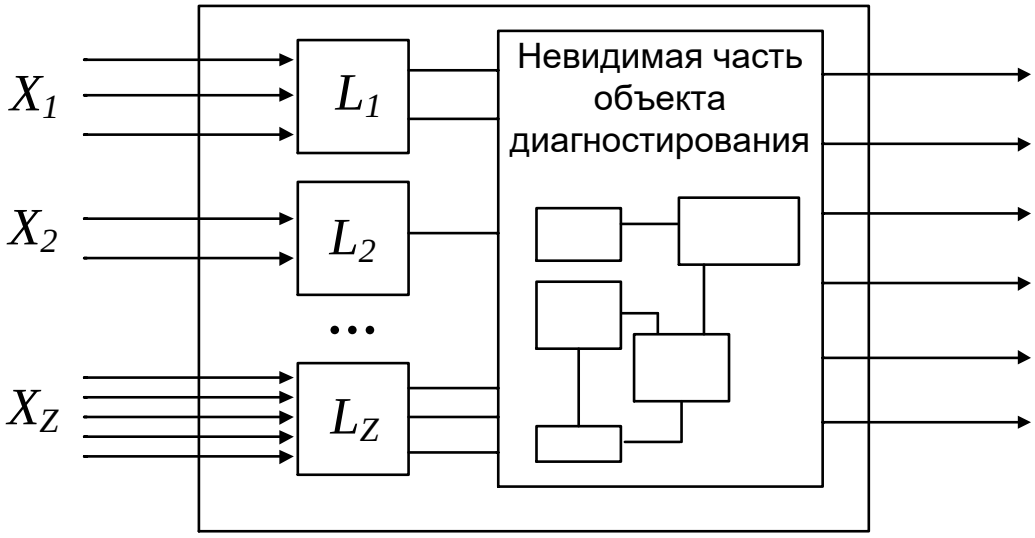

Рис.1 Представление диагностируемого изделия с z логическими интерфейсами

Как и в реальном диагностируемом изделии, так в его программной модели к «невидимой» части (рис.1) нет прямого доступа, т.е. на нее невозможно напрямую подать тестовые воздействия. Однако она на них реагирует опосредованно через краевые разъемы устройства [12].

Отдельный логический интерфейс характеризуется набором типовых операций, которые он выполняет, и совокупностью сигналов, поступающих в него (выдаваемых из него) при выполнении операций этого логического интерфейса [14]:

$$L_i = (K_i, O_i), \quad (9)$$

$$O_i = \left(O_{i,1}, O_{i,2}, \dots, O_{i,j}, \dots, O_{i,C_i}\right) \text{ для } j = 1, \dots, C_i\,, \quad (10)$$

$$K_i = \left(K_{i,1}, K_{i,2}, \dots, K_{i,j}, \dots, K_{i,C_i}\right), \quad (11)$$

$$K_{i,j} => O_{i,j}, \quad (12)$$

$$K_{i,j} = \left(K_{i,j,1}, K_{i,j,2}, \dots, K_{i,j,q}, \dots, K_{i,j,R_{ij}}\right) \text{ для } q = 1, \dots R_{ij}, \quad (13)$$

где O_i –множество типовых операций $O_{i,j}$, свойственных L_i ,

C_i – количество типовых операций $O_{i,j}$, свойственных L_i,

K_i – множество сигналов логического интерфейса L_i, поступающих (выдаваемых) на него при выполнении операций O_i .

При выполнении одной типовой операции $O_{i,j}$ логического интерфейса ЛИ L_i в него поступают (из него выдаются) подмножество сигналов $K_{i,j}$.

В качестве логических интерфейсов могут быть использованы счетчик, регистр, память, физические интерфейсы и т.п. Так, например, логический интерфейс «Счетчик» – L_i имеет следующее наполнение:

$$K_i = (\text{Сброс}, \text{Счет}+, \text{Счет}-, \text{Загрузка}_1, \dots, \text{Загрузка}_m), \quad (14)$$

$$O_i = (R, I, D, L), \quad (15)$$

где m – разрядность данных, загружаемых в счетчик,

R – операция обнуления счетчика,

I – операция увеличения содержимого счетчика на 1,

D – операция уменьшения содержимого счетчика на 1,

L – загрузка слова данных в счетчик.

Примером использования логических интерфейсов в реальном диагностируемом изделии может служить 8-ми разрядная память, подключенная к краевым разъемам изделия, реализованная при помощи 8 одноразрядных микросхем памяти. Если они объединены по линиям адреса, данных и управления соответственно, то тогда 8 соответствующих входных сигналов изделия могут быть привязаны к одному ЛИ «Память». Другой пример – в диагностируемом изделии имеется 6 микросхем 4-х разрядных счетчиков, связанных параллельной загрузкой и включенных последовательно. В этом случае они представляют собой 24-х разрядный счетчик, с возможностью параллельной загрузки. Если эти 6 микросхем счетчиков связаны с краевыми разъемами диагностируемого изделия, то тогда 24 соответствующих входных сигналов изделия могут быть привязаны к одному ЛИ «Счетчик».

Для каждого i-го логического интерфейса L_i, применяемого в объекте контроля, в процессе привязки уточняется перечень операций O_i, а также разрядность данных m и состав используемых сигналов K_i. Например, для логического интерфейса «Счетчик» в зависимости от реализации счетчика возможны различные варианты наполнения L_i. Некоторые наполнения L_i для 24-х разрядного счетчика приведены в таблице 1.

Таблица 1 – Варианты ЛИ «Счетчик» после привязке к диагностируемому изделию при m=24

K_c – сигналы L_c	**O_c – операции L_c**
Сброс, Счет +, Счет - , Загрузка$_1$,…, Загрузка$_{24}$	*R, I, D, L*
Сброс, Счет +, Загрузка$_1$,…, Загрузка$_{24}$	*R, I, L*
Сброс, Счет -, Загрузка$_1$,…, Загрузка$_{24}$	*R, D, L*

Пример представления диагностируемого изделия с шестью логическими интерфейсами (Clock - 1 ЛИ, Группа сигналов – 5 ЛИ) в программной реализации рассматриваемого способа представлен на рис.2.

На рис.2 логический интерфейс **CLOCK** («Тактовые импульсы») определяет последовательность тактовых импульсов, подаваемых в диагностируемое изделие. В примере на рис.2 этот интерфейс привязан ко входу диагноструемого изделия, на который подается сигнал **f_control**.

Логический интерфейс **GROUP_SIG** («Группа сигналов») задает периодическое изменение сигналов группы с заданным периодом, в которой происходит инкрементирование или декрементирование числового представления группы.

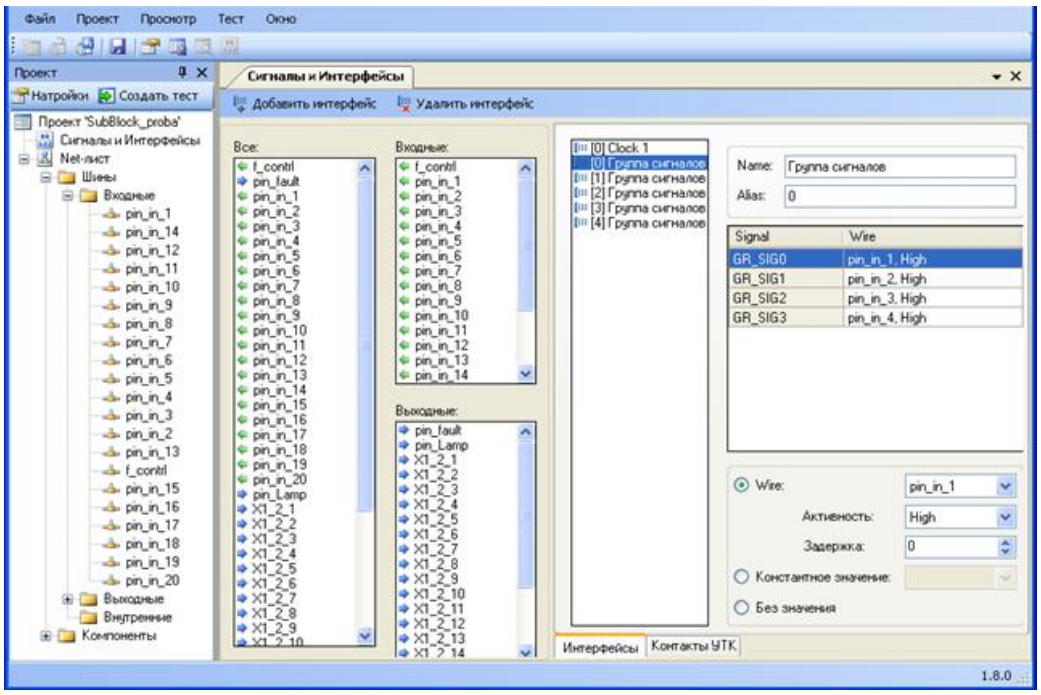

Рис. 2 Пример представления диагностируемого изделия с 6-ю логическими интерфейсами

Временные отрезки в логических интерфейсах измеряется в количестве сочетаний входных сигналов (векторах), одновременно подаваемых в диагностируемое изделие. Например, для подачи одного периода тактового импульса в диагностируемое изделие, необходимо передать два вектора (рис.3): один с логическим «0», а другой - с логической «1».

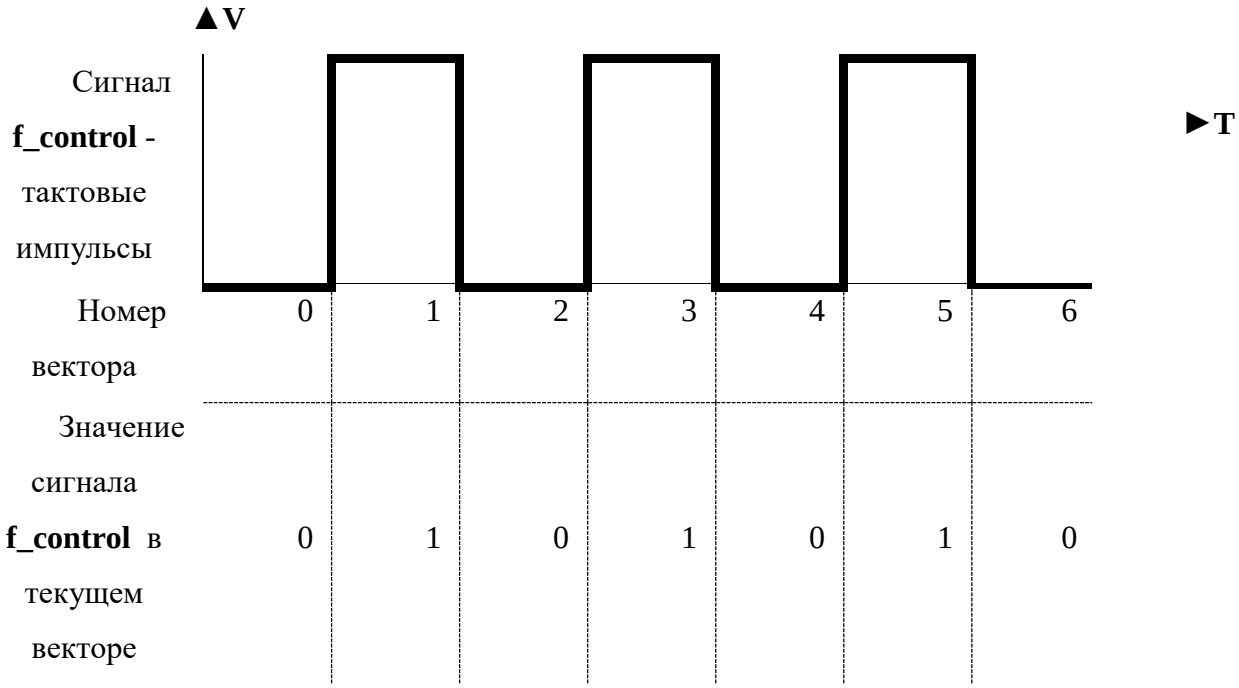

Рис. 3 Отражение значений сигнала **f_control**, к которому привязан логический интерфейс **CLOCK**, в вектора, подаваемые в диагностируемое изделие

После привязки входов к логическим интерфейсам (3) производится формирование векторов V, подаваемых в диагностируемое изделие:

$$V = (V_1, V_2, \dots, V_e, \dots, V_N) \text{ для } e = 1, \dots, N, \quad (16)$$

$$V_e = \left(v_{e,1}, v_{e,2}, \dots, v_{e,j}, \dots, v_{e,p}\right) \text{ для } j = 1, \dots, p, \quad (17)$$

где N – количество векторов создаваемого диагностического теста;

p – количество входов диагностируемого изделия.

Процесс формирования векторов V осуществляется на ЭВМ путем выполнения скрипта, задающего типы используемых логических интерфейсов L_i и последовательность выполнения их типовых операций O_i [12]. Пример скрипта, описывающего правила формирования входные воздействий диагностического теста в терминах логических интерфейсов представлен на рис.4.

```
    #GROUP_S        Переключение на 0-й ЛИ GROUP_SIG («Группа сигналов»)
IG 0
    Base 6          Начальное значение кода  группы
    Inc 1           Инкрементирование кода группы сигналов на 1
    Delay 2         Номер вектора, с которого начинается действие ЛИ
    Tcount 3        Кол-во  векторов, через которое производится инкрементирование
                    группы
    Tend -1         Инкрементирование производится в течении всего времени моделиро-
    //              вания
    #GROUP_S        Переключение на 1-й ЛИ GROUP_SIG («Группа сигналов»)
IG 1
```

Рис. 4 Пример описания входных воздействий теста в терминах ЛИ

Вектора V, формируемые после выполнения вышеприведенного скрипта (рис. 4) в программной реализации, представлены на рис.5.

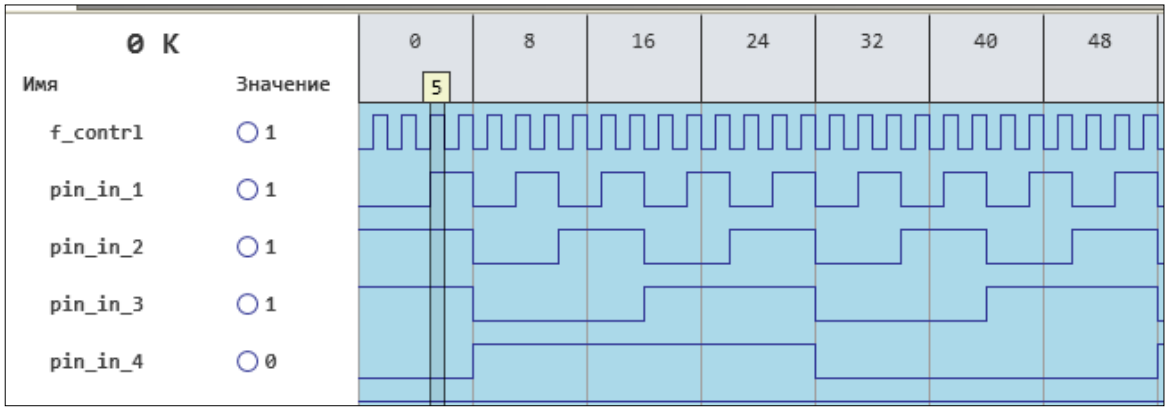

Рис.5 Входные воздействия теста, сформированные в результате применения логических интерфейсов.

Сформированные вектора V подают на входы X полученной программной модели диагностируемого изделия. Для каждого вектора V_e (10) определяют параметры сочетаний сигналов отклика на выходах программной модели диагностируемого изделия W_o и в характерных промежуточных точках между программными моделями составных частей изделия W_o:

$$W = (W_0, W_i), \tag{18}$$

$$W_0 = (W_{01}, W_{02}, \dots, W_{0e}, \dots, W_{0N}), \tag{19}$$

$$W_i = (W_{i1}, W_{i2}, \dots, W_{ie}, \dots, W_{iN}), \tag{20}$$

$$W = (W_1, W_2, \dots, W_e, \dots, W_N), \tag{21}$$

$$W_{0e} = \left(w_{0e,1}, w_{0e,2}, \dots, w_{0e,d_0}, \dots, w_{0e,T_0}\right) \text{ для } d_0 = 1, \dots, T_0, \tag{22}$$

$$W_{ie} = \left(w_{ie,1}, w_{ie,2}, \dots, w_{ie,d_i}, \dots, w_{ie,T_i}\right) \text{ для } d_i = 1, \dots, T_i, \tag{23}$$

$$W_e = \left(w_{0e,1}, w_{0e,2}, \dots, w_{0e,d_0}, \dots, w_{0e,T_0}, w_{ie,1}, w_{ie,2}, \dots, w_{ie,d_i}, \dots, w_{ie,T_i}\right) \tag{24}$$

где W – множество откликов эталонной модели диагностируемого изделия на подаваемые в нее векторов V,

T_o – количество выходов диагностируемого изделия,

T_i – количество характерных промежуточных точек между эталонными моделями составных частей изделия, с которых снимаются сигналов отклика,

$w_{oe,do}$ –сигнал, снимаемый с d_o – го выхода диагностируемого изделия в его e–ом отклике;

$w_{ie,di}$ –сигнал, снимаемый с d_i – ой характерной промежуточной точки диагностируемого изделия в его e–ом отклике.

Процесс формирования векторов *V*, их подачи в эталонную модель диагностируемого изделия и фиксация ее откликов *W* продолжается, пока не достигнуто требуемое значение предварительной эффективности создаваемого теста.

Полученные значения сигналов отклика *W*, вместе с параметрами соответствующих тестовых входных сигналов (векторов) *V*, заносят с помощью ЭВМ в базу данных.

Совокупность *(V,W)* заранее определенных входных воздействий *V* и соответствующих реакций *W* представляет собой контрольно-диагностический тест, который в дальнейшем используют для диагностики состояния диагностируемого изделия и диагностики неисправностей составных частей изделия [14].

Назначением контрольно-диагностических тестов является обнаружение и локализация максимально возможных неисправностей диагностируемых изделий. Это достигается обеспечением прохождения изменяющейся информации через максимально возможное количество цепей объектов контроля.

Эффективность каждого контрольно-диагностического теста определяется по результатам проведения отладки на реальном объекте контроля путем внесения в него искусственных неисправностей и определении реакции на эти неисправности анализируемого диагностического теста, а также по результатам реальной эксплуатации. Данная процедура кропотливая, монотонная, длительная, требует большого внимания, терпения и усидчивости. Это обуславливает необходимость сокращения временных и материальных затрат на проведение оценки эффективности диагностических тестов.

В качестве одного из путей решения этой проблемы предлагается определять степень покрытия информационных цепей объектов контроля до проведения испытаний. Проверка предварительной оценки эффективности теста

позволяет принять решение о готовности анализируемого теста к использованию при проведении испытаний для оценки его эффективности (качества) или об отправке анализируемого теста на доработку.

На основе описания внутренних частей объекта контроля формируют эквивалентную программную модель соединений, в ее разрывы включают заранее подготовленные поведенческие модели составных частей объекта контроля, для полученной программной модели объекта контроля формируют комбинации входных тестовых сигналов с заданными сочетаниями параметров сигналов и с заданными последовательностями подачи входных сигналов, для каждой комбинации входных тестовых сигналов определяют параметры сочетаний выходных сигналов, при этом:

- на входы сформированной программной модели объекта контроля задают соответствующие сочетания входных сигналов (входные воздействия) в соответствующей последовательности, заданной в оцениваемом контрольно-диагностическом тесте,
- для каждого сочетания задаваемых входных сигналов кроме первого определяют параметры сочетаний сигналов отклика на выходах программной модели объекта контроля и в характерных промежуточных точках между программными моделями составных частей объекта контроля, и, сравнивая сигналы отклика, полученные для предыдущего сочетания задаваемых входных сигналов, определяют изменения значений сигналов отклика,
- количество изменений значений параметров сигналов в каждой цепи программной модели объекта контроля подсчитывают с помощью ЭВМ и заносят в базу данных,
- на основании данных о количествах изменений значений параметров сигналов во всех информационных цепях программной модели объекта контроля после подачи в модели диагностируемого изделия всех входных воздействий из оцениваемого контрольно-

диагностического теста вычисляют предварительную эффективность этого теста (степени покрытия тестом информационных цепей диагностируемого изделия) по формуле:

$$K = {}^{N_{\text{изм}}}\!/_{N} \cdot 100\% \qquad (25)$$

где N – количество цепей, доступ к которым осуществляется через выходы программной модели объекта контроля и характерные промежуточные точки между программными моделями составных частей изделия,

$N_{\text{изм}}$ – количество активированных из N цепей (в которых произошло хотя бы одно изменение значений параметров сигналов),

– для каждого выхода программной модели объекта контроля и каждой характерной промежуточной точки между программными моделями составных частей изделия визуально отображают признак изменения информации (того, изменилось ли состояние) параметров сигналов в них за время прохождения оцениваемого контрольно-диагностического теста,

– на основании значения предварительной эффективности *K* оцениваемого контрольно-диагностического теста, а также визуального отображения признаков изменения параметров сигналов на выходах программной модели объекта контроля и характерных промежуточных точках между программными моделями составных частей изделия принимают предварительное решение о достаточной эффективности оцениваемого контрольно-диагностического теста, в результате которого оцениваемый диагностический тест отправляют на доработку или направляют на экспериментальную проверку.

Применение логических интерфейсов и генерация скрипта при формировании тестовых программ позволяют существенно сократить время на разработку контрольно-диагностических тестов, сократить трудозатраты, а также снизить требования к квалификации разработчика.

Таким образом, можно выделить следующие этапы формирования контрольно-диагностического теста для диагностируемого изделия:

1. Анализ объекта контроля (ОК);
2. формирование модели функционирования всего ОК, получение информации о связях компонентов ОК;
3. разработка моделей функционирования компонентов ОК на языке Verilog HDL;
4. привязка логических интерфейсов к ОК;
5. генерация теста (формирование входных воздействий);
6. моделирование тест-программы (получение откликов модели);
7. предварительная оценка эффективности теста;
8. в случае получения теста неудовлетворительного качества переход на шаг 5;
9. отладка тестовой программы на реальном объекте контроля;
10. в случае некорректного прогона теста на реальном объекте контроля (останов по браку) переход на шаг 3.

Рассмотренный способ формирования контрольно-диагностических тестов был применен мной к более, чем 10 объектам контроля. Более подробно все этапы формирования диагностического теста рассмотрены на примере объекта контроля ИЗД.032.677 в практической части.

5 Описание практической части

5.1 Эмуляция цифрового объекта контроля

5.1.1 Анализ объекта контроля

Объект контроля представляет собой цифровую ячейку, изображение которой представлено на рис.6.

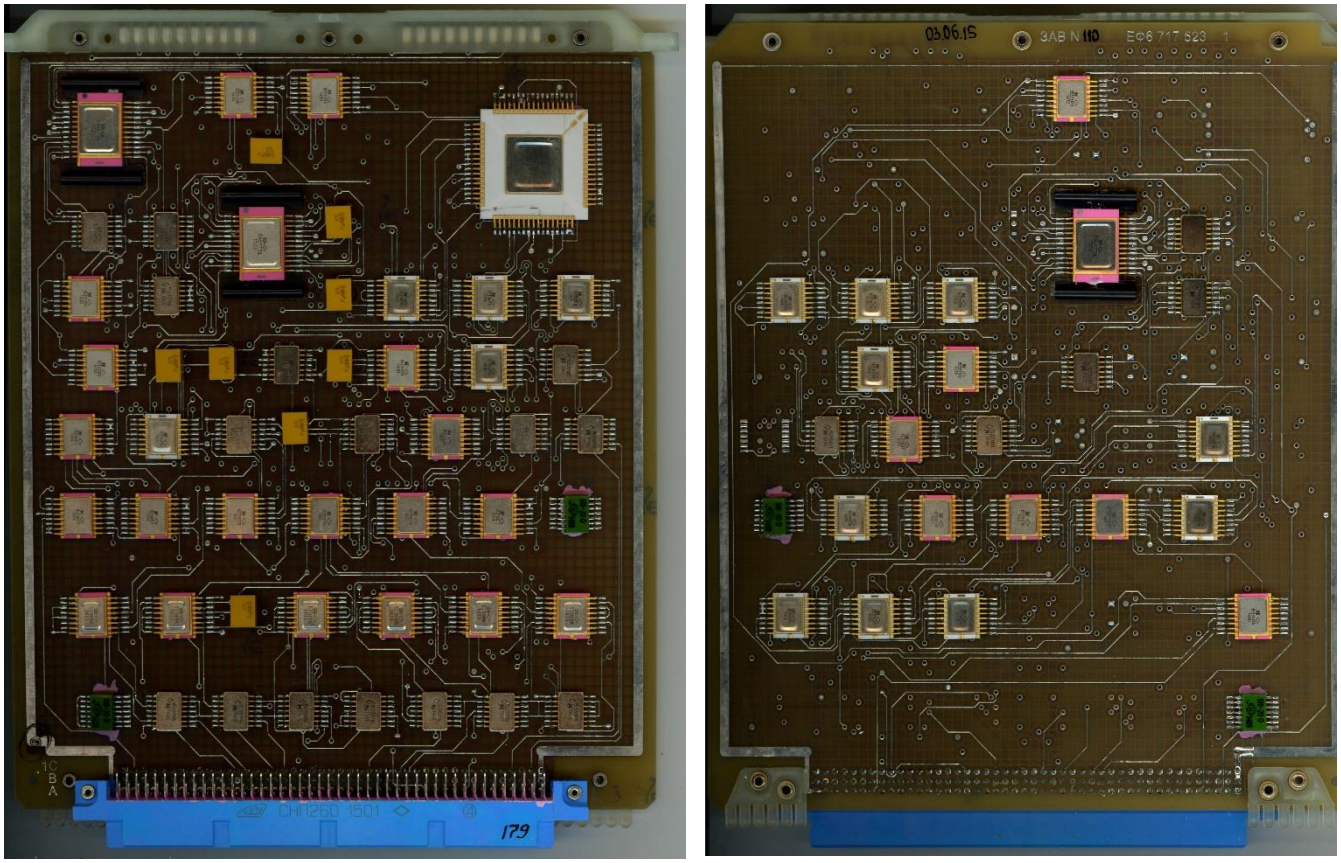

Рис.6 Изображение цифровой ячейки ИЗД.032.677

Фрагмент схематического изображения диагностируемого изделия со всеми компонентами и связями между ними приведен на рис.7.

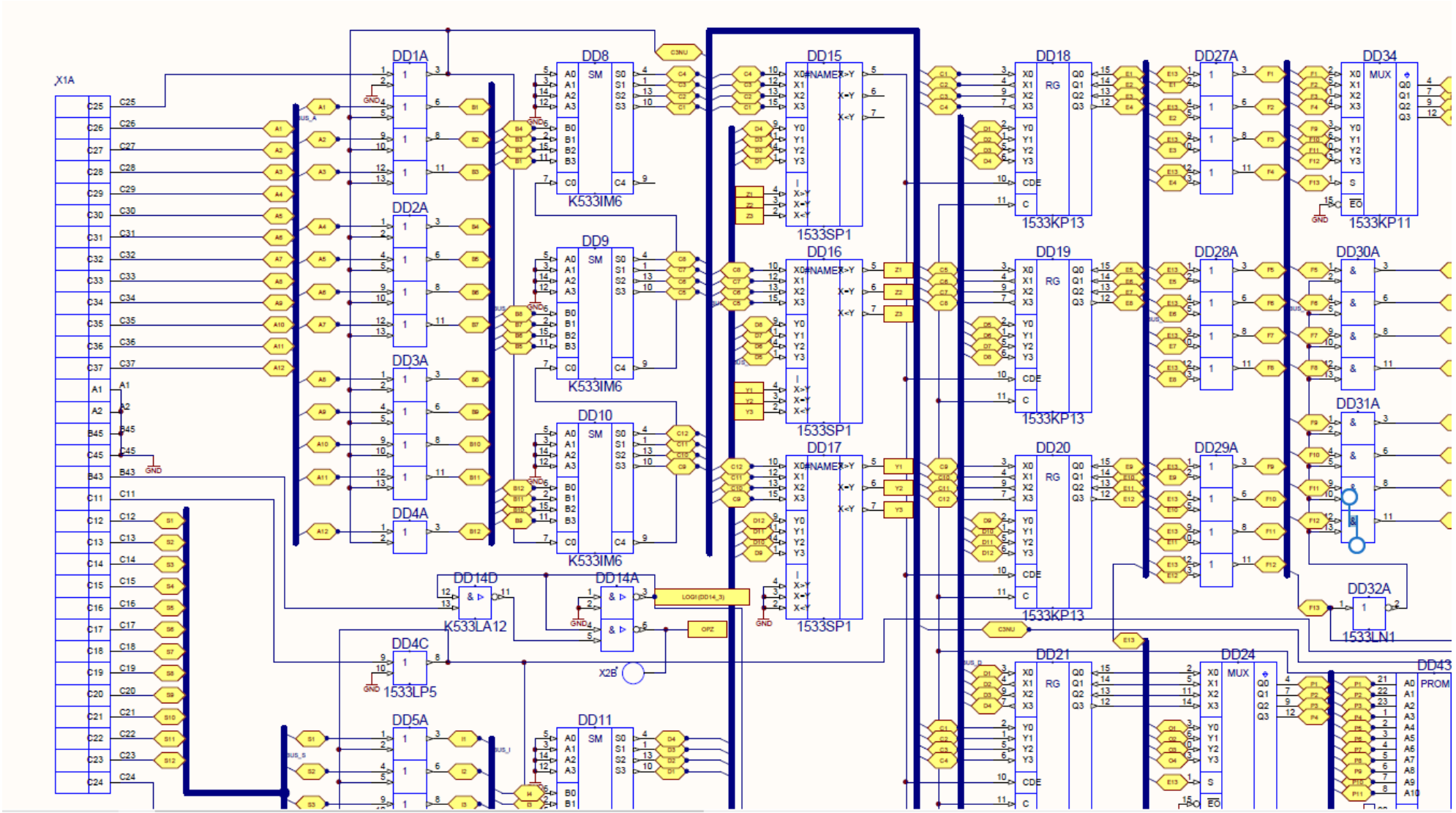

Рис.7 Фрагмент схематического представление цифрового объекта контроля ИЗД.032.677

Под цифровым электронным устройством подразумевается набор электронных компонентов, каждый из которых функционирует по собственному алгоритму, при этом алгоритм функционирования электронного устройства определяется алгоритмами функционирования компонентов, связями между модулями и временными последовательностями входных сигналов. Поэтому необходимо в первую очередь проанализировать все компоненты устройства. Перечень элементов объекта контроля представлен в таблице 2.

Таблица 2 – Перечень элементов объекта контроля.

	Наименование микросхемы	Элемент на схеме
1	1533ЛП5	D1, D2, D3, D4, D5, D6, D7
2	533ИМ6	D8, D9, D10, D11, D12, D13
3	533ЛА12	D14
4	1533СП1	D15, D16, D17
5	1533КП13	D18, D19, D20, D21, D22, D23
6	533ЛЛ1	D27, D28, D29
7	1533ЛИ1	D30, D31, D33
8	1533ЛН1	D32, D61
9	1533КП11	D24, D25, D26, D34, D35, D36, D37, D38, D39, D40, D41, D42
10	1533ИЕ7	D47, D48, D49
11	1533ТМ9	D50, D51, D55, D56, D57, D62, D63, D70
12	1533ТМ2	D52
13	1533ЛР11	D53
14	1802ВР4	D54
15	556РТ7А	D43, D44, D58
16	1533КП7	D59, D60, D65
17	533ИП5	D45, D46, D64

Наибольшую сложность представляют собой микросхемы 1802 ВР4 – умножитель 12*12 и 556РТ7А – программируемое постоянное запоминающее устройство.

При анализе объекта контроля по параметрам работы компонентов (указанным в документации на каждый компонент) делается вывод о частоте тестирования – для объекта контроля ИЗД.032.677 частота тестирования составляет 1МГц.

5.1.2 Формирование модели функционирования объекта контроля, получение информации о связях компонентов

Формирование модели функционирования объекта контроля в целом происходит за счет оцифровки схемы. Оцифровка схемы представляет собой процесс создания схемы устройства в электронном виде с помощью мультиплатформенной среды проектирования Altera Quartus II [10]. При оцифровке схемы (рис.8) используется стратегия восходящего проектирования, так как объект контроля имеет детальное структурное описание (как правило — принципиальная схема на микросхемах средней степени интеграции).

Информация о связях компонентов диагностируемого изделия, которая представляет собой поведенческую модель объекта контроля (Netlist), автоматически генерируется системой при помощи программных возможностей среды проектирования Altera Quartus II на основе электронного представления схемы, затем загружается в базу данных системы автоматического проектирования «SimTest».

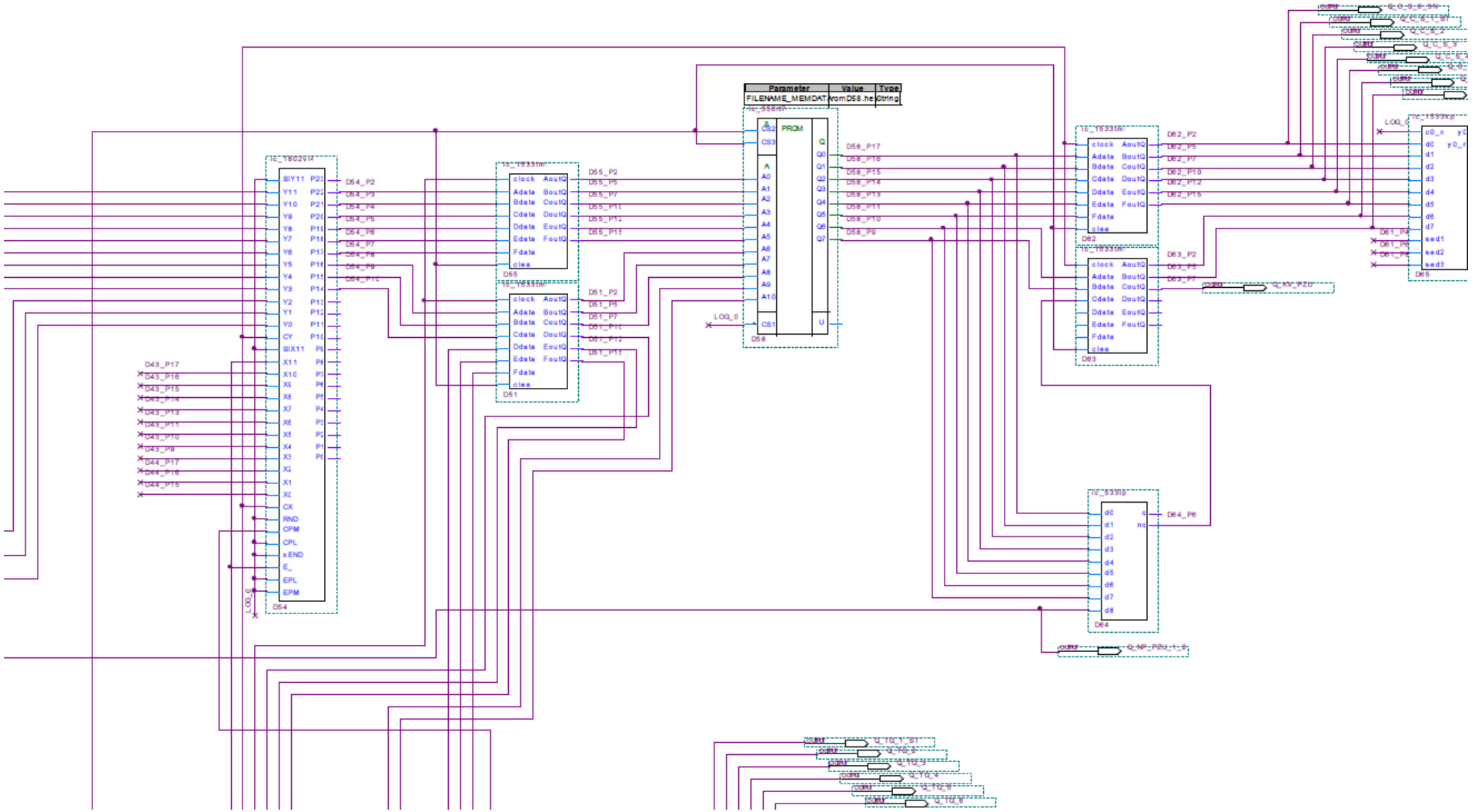

Рис.8 Фрагмент оцифрованной при помощи среды проектирования Altera Quartus II схемы ИЗД.032.677.

5.1.3 Разработка моделей функционирования компонентов объекта контроля на языке Verilog HDL

Модель диагностируемого изделия реализуется на одном из HDL-языков описания электронных устройств (Hardware Description Language). В качестве языка моделирования в рассматриваемой системе используется Verilog [7], поскольку он синтаксически проще других HDL-языков и для него существует ряд свободно распространяемых систем для моделирования. Все этапы разработки тестовой программы поддерживаются системой за исключением оцифровки схемы и физической отладки теста на установке тестового контроля.

Поскольку объект контроля состоит из различных компонентов, связанных определенным образом, модель устройства в целом должна включать модели каждого компонента на HDL-языке Verilog и учитывать связи между ними.

Ниже приведены описания компонентов объекта контроля, а также их программные модели на Verilog-HDL, также загружаемые в базу данных системы автоматического проектирования «SimTest»:

5.1.3.1 1533ЛП5 – четыре двухвходовых элемента логических «Исключающее ИЛИ».

Микросхема 1533ЛП5 содержит четыре независимых логических элемента «Исключающее ИЛИ», выполняющих Булеву функцию $Y = \overline{D1}D2 + D1\overline{D2}$.

В программной реализации элемент выглядит следующим образом:

```
module ic_1533lp5(i1, i2, o);

input i1, i2;
output o;

assign o = i1 ^ i2;

endmodule
```

5.1.3.2 533ИМ6 – 4-битный двоичный сумматор с быстрым переносом.

Микросхема 533ИМ6 предназначена для суммирования двух 4-битных двоичных слов. Суммирование происходит побитно, на выходе C4 появляется напряжение высокого уровня, когда на всех выходах присутствует логическая «1».

Фрагмент программной реализация компонента:

```
     module ic_533IM6 ( A0, A1, A2, A3, B0, B1, B2, B3, C0, S0, S1,
S2, S3, C4);
     input C0, A0, A1, A2, A3, B0, B1, B2, B3;
     output S0, S1, S2, S3, C4;

     ic_74S283 add (C0, {A3,A2,A1,A0}, {B3,B2,B1,B0}, {S3,S2,S1,S0},
C4);

     endmodule
```

Полная программная реализация компонента в приложении 1.

5.1.3.3 533ЛА12 – 4 логических элемента 2И-НЕ.

Микросхема представляет собой четыре идентичных логических элемента со стандартными активными выходами, выполняющих Булеву функцию $Y = \overline{D1D2}$ в положительной логике.

Программная реализация компонента:

```
`timescale 1 ns/1 ns
module ic_1533la12(i1, i2, o);

input i1, i2;
output o;

reg _o;

always @* begin
     _o = ~(i1 & i2);
end

assign o = _o;

endmodule
```

5.1.3.4 1533СП1 – 4-х разрядная схема сравнения чисел.

Микросхема 1533СП1 предназначена для сравнения 4-разрядных двоичных чисел, представленных в прямом коде. Сравнение проводится, начиная со

старших разрядов. Если они различны, то эти разряды определяют результат сравнения, в противном случает проводится сравнение последующих младших разрядов и т.д [8].

Элемент имеет необходимые средства для наращивания разрядности сравниваемых чисел без использования дополнительных внешних логических элементов. Подобная ситуация представлена на схеме ИЗД.032.677 – три последовательно включенных схемы сравнения D15, D16, D17. При этом выходы AmB, AlB и AeB микросхемы, производящей сравнение младших разрядов (D17), соединяются с соответствующими входами AmBi, AlBi и AeBi микросхемы, производящей сравнение более старших разрядов числа (D16). На вход AeBi микросхемы, производящей сравнение самых младших разрядов числа, должен быть подан высокий уровень напряжения – логическая «1».

Программная модель 1533СП1:

```
module ic_1533sp1(a0, a1, a2, a3, b0, b1, b2, b3, AmBi, AeBi,
AlBi, AmB, AeB, AlB);

      // AmB ~ A [m]ore than B
      // AeB ~ A [e]qual B
      // AlB ~ A [l]ess than B

input a0, a1, a2, a3, b0, b1, b2, b3, AmBi, AeBi, AlBi;
output reg AmB, AeB, AlB;

always @* begin
      if({a3, a2, a1, a0} == {b3, b2, b1, b0}) begin
            AeB <= AeBi;
            if(AeBi)
                  {AmB, AlB} <= 2'b00;
            else
                  case ({AmBi, AlBi})
                        2'b00 : {AmB, AlB} <= 2'b11;
                        2'b11 : {AmB, AlB} <= 2'b00;
                        default : {AmB, AlB} <= {AmBi, AlBi};
                  endcase
      end
      else if({a3, a2, a1, a0} > {b3, b2, b1, b0})
            {AmB, AeB, AlB} <= 3'b100;
      else // ({a3, a2, a1, a0} < {b3, b2, b1, b0})
            {AmB, AeB, AlB} <= 3'b001;
end

endmodule
```

5.1.3.5 1533КП13 – четыре двухвходовых мультиплексора с запоминанием.

При низком уровне напряжения на входе выбора слова микросхемы 1533КП13 на внутренние входы регистра поступает слово A1, B1, C1, D1, при высоком уровне – слово A2, B2, C2, D2. По отрицательному фронту сигнала синхронизации выбранное слово записывается в регистр, передается на выход.

```
module ic_1533kp13(A1,B1,C1,D1,A2,B2,C2,D2,SE,C,Q1,Q2,Q3,Q4);

input A1, B1, C1, D1, A2, B2, C2, D2, SE, C;
output Q1, Q2, Q3, Q4;
reg  Q4;
reg  Q3;
reg  Q2;
reg  Q1;

always@(negedge C)
begin
     begin
     Q1 <= A1 & (~SE)| SE & A2;
     Q2 <= B1 & (~SE)| SE & B2;
     Q3 <= C1 & (~SE)| SE & C2;
     Q4 <= D1 & (~SE)| SE & D2 ;

     end
end

endmodule
```

5.1.3.6 533ЛЛ1 – четыре логических элемента 2ИЛИ.

Микросхема 533ЛЛ1 представляет собой четыре логических элемента со стандартными активными выходами, выполняющие Булеву функцию $Y = \overline{D1} + \overline{D2}$.

```
`timescale 1 ns/1 ns
module ic_533ll1(i1, i2, o);

input i1, i2;
output o;

reg _o;

always @* begin
     _o = #0 (i1 | i2);
end

assign o = _o;

endmodule
```

5.1.3.7 1533ЛИ1 – четыре логических элемента 2И.

Микросхема 1533ЛИ1 представляет собой четыре идентичных логических элемента со стандартными активными выходами, выполняющие Булеву функцию $Y = D1D2$.

Программная реализация микросхемы 1533ЛИ1:

```
module ic_1533LI1(
input in0,in1,
output     out);

assign     out = (in0 & in1);

endmodule
```

5.1.3.8 1533ЛН1 – шесть логических элементов НЕ.

Микросхема 1533ЛН1 представляет собой шесть идентичных логических элементов со стандартными активными выходами, выполняющих Булеву функцию $Y = \overline{D}$.

Программная реализация микросхемы 1533ЛН1:

```
`timescale 1 ns/1 ns
module ic_1533ln1(i, o);

input i;
output o;

reg o;

always @*
      o = ~i;

endmodule
```

5.1.3.9 1533КП11 – четырехразрядный селектор 2-1 с тремя устойчивыми состояниями на выходе.

Микросхема 1533КП11 представляет собой четырехразрядный селектор-мультиплексор 1 из 2 без инверсии входов и с тремя состояниями на выходах. При напряжении высокого уровня на входе управления третьим состоянием EZ_n выходы микросхемы переводятся в высокоимпедансное (третье) состояние, а при низком уровне – на выходы Q передается информация с со-

ответствующих входов D, определенных логическим состоянием входа выбора SE. Наличие выходов с тремя состояниями позволяет использовать микросхему для работы непосредственно на магистраль [8].

Программная реализация микросхемы 1533КП11:

```
     module ic_1533kp11(d00, d01, d02, d03, d10, d11, d12, d13, se,
ez, y0, y1, y2, y3);

     input d00, d01, d02, d03, d10, d11, d12, d13, se, ez;
     output y0, y1, y2, y3;

     wire [3:0] d;

     assign d = se ? {d13, d12, d11, d10} : {d03, d02, d01, d00};
     assign {y3, y2, y1, y0} = ez ? 4'bz : d;

     endmodule
```

5.1.3.10 533ИЕ7 – четырехразрядный двоичный реверсивный счетчик.

Микросхема 1533ИЕ7 представляет собой четырехразрядный двоичный реверсивный счетчик синхронного типа. Импульс высокого уровня напряжения по входу r устанавливает выходы счетчика в исходное состояние – в низкий уровень напряжения на счетных выходах [8]. Для предварительной установки микросхемы в определенное состояние необходимо на информационные входы подать соответствующие уровни напряжения, а на вход стробирования предварительной записи подать отрицательный импульс напряжения. Для осуществления прямого счета необходимо на вход down подать высокий уровень напряжения, а на вход прямого счета up – положительные импульсы. Счет будет осуществляться от того числа, которое было предварительно записано в счетчик с информационных входов D. После заполнения счетчика выходы устанавливаются в состояние высокого уровня напряжения, а на выходе прямого переноса появляется отрицательный импульс переноса счета в старший разряд (в случае, если счетчики подключены последовательно, как и приведено на схеме – D47, D48, D49). Аналогично счетчик 1533ИЕ7 работает в режиме обратного счета [9].

Программная реализация счетчика 1533ИЕ7:

```
`timescale 1 ns/1 ns
module ic_1533ie7(up, down, wr, d0, d1, d2, d3, r, cr, br, q0, q1, q2,
q3);

      input up, down, wr, d0, d1, d2, d3, r;
      output cr, br;
      output q0, q1, q2, q3;

      parameter INITIAL_VAL = 4'b0000;

      reg [3:0] out;
      wire count, u, d;

      initial    out = INITIAL_VAL;

      assign #1 u = up;
      assign #1 d = down;
      assign count = up & down;

      assign q0 = out[0];
      assign q1 = out[1];
      assign q2 = out[2];
      assign q3 = out[3];

      always @(posedge count or posedge r or negedge wr)
            if(r)
                  out <= 4'b0000;
            else if(!wr)
                  out <= {d3, d2, d1, d0};
            else if(!u & d)
                  out <= out + 4'b0001;
            else if(!d & u)
                  out <= out - 4'b0001;


      assign cr = ~(q0 & q1 & q2 & q3 & !up);
      assign br = ~(!q0 & !q1 & !q2 & !q3 & !down);

endmodule
```

5.1.3.11 1533ТМ9 – шесть синхронных D-триггеров.

Микросхема 1533ТМ9 содержит шесть триггеров D-типа с прямыми выходами, предназначена для создания устройств память цифровой автоматики широкого применения.

Отличительная особенность микросхемы 1533ТМ9 – наличие общих для всех триггеров входа синхронизации C и входа сброса R_n. Тактирование осуществляется за счет переднего фронта синхросигнала C, а установка прямых выходов в состояние низкого уровня напряжения – низким уровнем на входе сброса R_n [9].

Программная реализация триггера 1533ТМ9:

```
module ic_1533tm9 (clock, Adata, Bdata, Cdata, Ddata, Edata, Fdata,
AoutQ, BoutQ, CoutQ, DoutQ, EoutQ, FoutQ, clear);

   input       Adata, Bdata, Cdata, Ddata, Edata, Fdata, clock, clear;
   output      AoutQ, BoutQ, CoutQ, DoutQ, EoutQ, FoutQ;

   reg         Aout = 1'b0;
   reg         Bout = 1'b0;
   reg         Cout = 1'b0;
   reg         Dout = 1'b0;
   reg         Eout = 1'b0;
   reg         Fout = 1'b0;

   wire        AoutQ;
   wire        BoutQ;
   wire        CoutQ;
   wire        DoutQ;
   wire        EoutQ;
   wire        FoutQ;


   assign AoutQ = clear ? Aout : 1'b0;
   assign BoutQ = clear ? Bout : 1'b0;
   assign CoutQ = clear ? Cout : 1'b0;
   assign DoutQ = clear ? Dout : 1'b0;
   assign EoutQ = clear ? Eout : 1'b0;
   assign FoutQ = clear ? Fout : 1'b0;


   always @(posedge clock or negedge clear) begin
      if (!clear) begin
         Aout <= 1'b0;
         Bout <= 1'b0;
         Cout <= 1'b0;
         Dout <= 1'b0;
         Eout <= 1'b0;
         Fout <= 1'b0;
      end
      else if (clear) begin
         Aout <= Adata;
         Bout <= Bdata;
         Cout <= Cdata;
         Dout <= Ddata;
         Eout <= Edata;
         Fout <= Fdata;
      end
   end
     endmodule
```

5.1.3.12 1533ТМ2 – два синхронный D-триггера с дополняющими выходами.

Микросхема 1533ТМ2 представляет собой два независимых триггера D-типа, срабатывающих по переднему фронту тактового сигнала c.

Низкий уровень напряжения на входах установки s или сброса r устанавливают выходы триггера в соответствующее состояние в зависимости от состояний на входе тактирования c и информационном входе d. При наличии высокого напряжения на входах установки s и сброса r для корректной работы триггера требуется предварительная установка информации по воду данных d относительно переднего фронта сигнала c и соответствующая выдержка информации после подачи положительного фронта синхросигнала [8].

Программная реализация триггера 1533ТМ2:

```
`timescale 1 ns/100 ps
module ic_1533tm2(s, d, c, r, q, nq);

      input s, c, d, r;
      output reg q, nq;

      initial begin
            q = 0;
            nq = 1;
      end

      always @(posedge c or negedge r or negedge s) begin
            if(!r) begin
                  q <= 0;
                  nq <= 1;
            end
            else if(!s) begin
                  q <= 1;
                  nq <= 0;
            end
            else begin
                  q <= d;
                  nq <= ~d;
            end
      end

endmodule
```

5.1.3.13 1533ЛР11 – два логических элемента: 2-2И-2ИЛИ-НЕ и 3-3И-2ИЛИ-НЕ.

Микросхема представляет собой два логических элемента, выполняющими Булевы функции $Y1 = \overline{1D1 * 1D2 * 1D3 + 1D4 * 1D5 * 1D6}$ и $Y2 = \overline{2D1 * 2D2 + 2D3 * 2D4}$.

Программная реализация микросхемы 1533ЛР11:

```
module ic_533lr11(i1, i2, i3, i4, i5, i6, o);

      input i1, i2, i3, i4, i5, i6;
      output o;

      assign o = ~((i1 & i2 & i3 )||(i5 & i4 & i6));

endmodule
```

5.1.3.14 1802ВР4 – умножитель на 12 разрядов.

а) Назначение.

Большая интегральная схема 1802 ВР4 представляет собой умножитель на 12 разрядов, которая предназначена для применения в радиоэлектронной аппаратуре специального назначения в качестве быстродействующего устройства для перемножения двенадцатиразрядных операндов.

Каждый из операндов может быть либо кодом (числом без знака), либо числом со знаком. В последнем случае операнд представляется в дополнительном коде. Числа могут быть как целыми, так и меньше единицы.

На выходе умножителя вырабатывается произведение двойной точности – 24 разряда, которое может быть округлено до 12 разрядов (включая знаковый разряд). При умножении чисел со знаков в дополнительном коде произведение получается в дополнительном коде. При действии над числами со знаком предусмотрена возможность присвоения знака произведения младшей части произведения.

Умножитель является устройством модульного типа, обеспечивающим построение умножителей с любой разрядностью операндов.

Умножитель может быть использован для построения быстродействующих процессов цифровой обработки сигналов, реализующих алгоритм быстрого преобразования Фурье, цифровую фильтрацию и т.д. Возможно применение умножителя также в специализированных и универсальных цифровых электронных вычислительных машинах. Наличие регистров сомножителей и произведения, а также возможность управления «прозрачностью» последнего делает возможным более шибкое использование умножителя в конвейерных системах [8].

Применение на выходе умножителя буферных схем с тремя состояниями позволяет объединить выходы нескольких умножителей в одну шину произведения (например, при уменьшении времени умножения массивов чисел методом мультиплексирования).

б) Принцип работы.

Условное графическое обозначение представлено на рис. 9, назначение выводов в табл.3.

Таблица 3 – Назначение выводов большой интегральной микросхемы 1802 ВР4.

Вывод	Назначение
58, 59, 60	Питание +5 В
51-56, 61, 62, 64-67	Входы множителя, разряды 0-11 (DY0-DY11)
17, 34, 63	Общий 0 В
68	Вход знака весового коэффициента старшего разряда множителя (SIY11)
1-12, 22, 23	Выходы произведения, разряды 23-0 (DP23-DP0)
13	Вход синхронизации регистра старшей части произведения (CPM)
14	Вход синхронизации регистра младшей части произведения (CPL)
15	Вход управления сдвигом вправо старшей части произведения (E –>)

16	Вход управления «прозрачностью регистров произведения (CORGP)
20	Вход управления входными буферными каскадами старшей части произведения (EZPM)
21	Вход управления выходными буферными каскадами младшей части произведения (EZPL)
25-46	Входы множимого, разряды 0-11 (DX0-DX11)
47	Вход синхронизации регистра множимого (CX)
48	Вход синхронизации регистра множителя (CY)
49	Вход округления (RND)
50	Вход знака весового коэффициента старшего разряда множимого (SIX11)

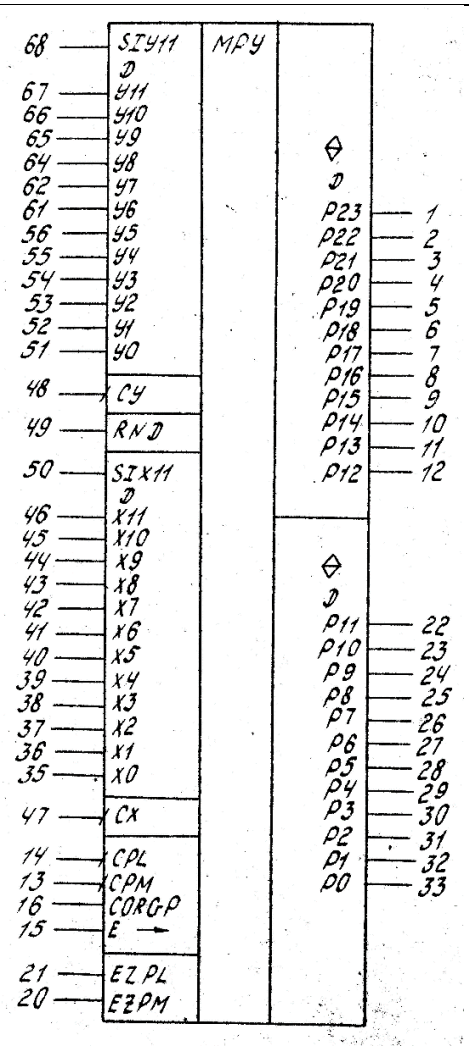

Рис.9 Условное графическое изображение БИС 1802 ВР4.

Структурная схема большой интегральной схемы приведена на рисунке 10.

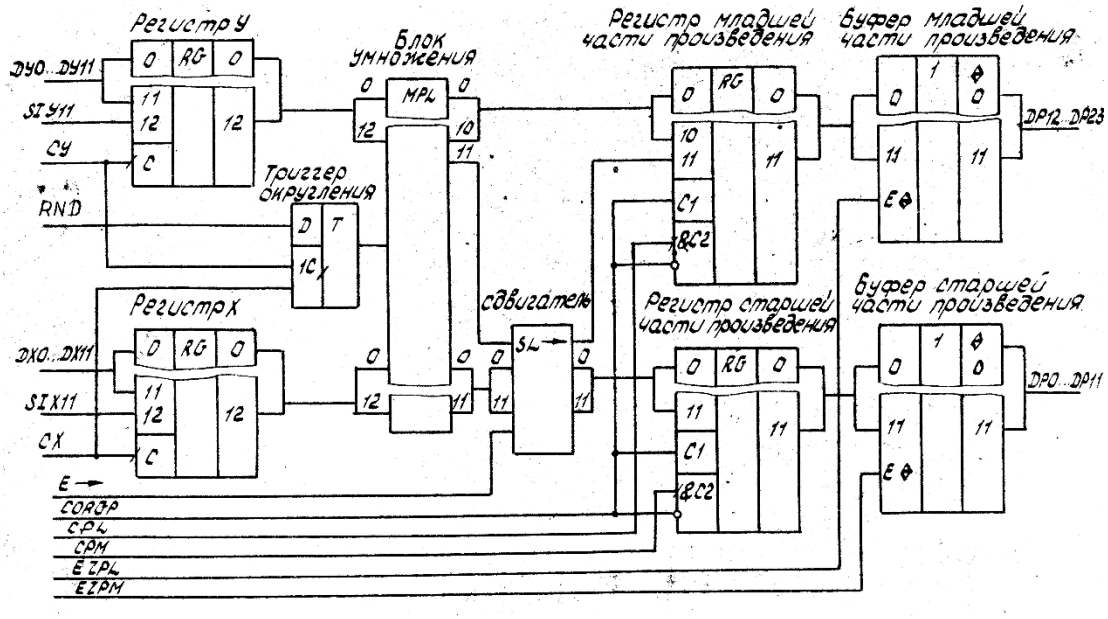

Рис. 10 Структурная схема большой интегральной схемы 1802 ВР4.

Большая интегральная схема 1802 ВР4 включает в себя ряд устройств, в том числе: регистр множимого (регистр X), регистр множителя (регистр Y), триггер округления, блок умножения, сдвигатель, регистры младшей части и старшей части произведения и выходные буферные каскады младшей и старшей части произведения.

Регистры X и Y выполнены на D-триггерах с одним тактирующим входом. Запись в них осуществляется по переднему фронту сигналов CX, CY соответственно. Помимо 12 разрядов сомножителей в регистры X и Y заносятся признаки множимого и множителя SIX11 и SIY11, которые имеют значение «1» в том случае, если сомножитель – число со знаком, и «0», если сомножитель – число без знака. Другими словами, SIX11 и SIY11 – знаки весового коэффициента старшего разряда числа: при значении «1» - минус, при «0» - плюс.

По переднему фронту одного из сигналов CX или CY в триггер округления записывается сигнал RND, по которому производится округление произведения до 12 разрядов в случае RND = 1.

Блок умножения представляет собой комбинационную матрицу, в которой формируются частичные произведения от поразрядного умножения множимого на множитель, суммируются с соответствующими весами и результат корректируется при действии над числами со знаком.

Операция округления выполняется одновременно с суммированием частичных произведений путем прибавления единицы в старший разряд отбрасываемой части, т.е. дополнительной задержки не вносит.

При действии над числами со знаком предусмотрена возможность присвоения знака произведения его младшей части. Для этого на входе E–> должно быть напряжение низкого уровня, т.е. E–>=0, при этом DP11=DP23. Следует иметь ввиду возможность получения неправильного результата, если E–>=0 из-за переполнения разрядной сетки при умножении максимальных по модулю отрицательных чисел [9].

При перемножении кодов или смешанном перемножении всегда на вход E–> необходимо подавать напряжение высокого уровня для получения правильного результата.

С помощью сдвигателя SL, управляемого сигналом E–> произведение P в соответствующем формате подается на регистры младшей и старшей части произведения (LSP и MSP соответственно).

Запись в регистр произведения происходит по переднему фронту сигналов CPL, CPM соответственно в случае CORGP = 0. Если CORGP = 1, сигналы CPL и CPM блокируются, D-триггеры регистров становятся потенциальными триггерами и информация от входов регистров постоянно проходит на выходы (регистры как бы исключены – «прозрачны»).

Выходные буферные каскады управляются сигналами EZPL (младшая часть произведения) и EZPM (старшая часть). Каскады находятся в выключенном (третьем) состоянии, когда управляющий сигнал равен «1».

Для построения умножителей с большей разрядной сеткой используется принцип разбиения слов операндов на части. Поскольку при перемножении

чисел со знаком или смешанном произведении некоторые частичные произведения будут так же со знаком, то старший разряд таковых имеет отрицательный вес и при умножении частичных произведений должен расшириться. Таким образом, для перемножения N-разрядных операндов необходимо:

1) разбить операнды на K = N/12 частей по 12 разрядов;
2) реализовать получение частичных произведений с помощью умножителя 1802 ВР4;
3) сложить частичные произведения с учетом их весов и наличия или отсутствия у них знака.

14.3 Программная реализация 1802 ВР4:

```
module ic_1802vr4 (
      input SIY11, Y11, Y10, Y9, Y8, Y7, Y6, Y5, Y4, Y3, Y2, Y1, Y0, CY,
      input SIX11, X11, X10, X9, X8, X7, X6, X5, X4, X3, X2, X1, X0, CX,
      input RND, CPM, CPL, xEND, E_, EPL, EPM,
      output P23, P22, P21, P20, P19, P18, P17, P16, P15, P14, P13, P12, P11,
P10, P9, P8, P7, P6, P5, P4, P3, P2, P1, P0
);

      wire [11:0] Xn;
      wire [11:0] Yn;
      reg [23:0] R;
      assign Xn = { X11, X10, X9, X8, X7, X6, X5, X4, X3, X2, X1, X0 };
      assign Yn = { Y11, Y10, Y9, Y8, Y7, Y6, Y5, Y4, Y3, Y2, Y1, Y0 };

      assign { P23, P22, P21, P20, P19, P18, P17, P16, P15, P14, P13, P12} =
EPM ? 12'bz : ( xEND ? TR[23:12] : R[23:12] );
      assign { P10, P9, P8, P7, P6, P5, P4, P3, P2, P1, P0 } = EPL ? 11'bz : (
xEND ? TR[10:0] : R[10:0]);

      reg signed [11:0] DX;
      reg SX;
      reg signed [11:0] DY;
      reg SY;
      reg DRND;
      reg signed [23:0] TR;
      reg signed [23:0] TX;
      reg signed [23:0] TY;

      wire signed [23:0] DX11;
      wire signed [23:0] DY11;

      assign DX11 = !SX ? { DX[11], 11'b0 } : -{ DX[11], 11'b0 };
      assign DY11 = !SY ? { DY[11], 11'b0 } : -{ DY[11], 11'b0 };

      always @(posedge CX) DX <= Xn;
      always @(posedge CX) SX <= SIX11;
      always @(posedge CY) DY <= Yn;
      always @(posedge CY) SY <= SIY11;
      always @(posedge CX, posedge CY) DRND <= RND;
      always @(posedge CPL, posedge CPM, negedge xEND) R <= TR;
      always @( DX, DX11 ) TX <= DX[10:0] + DX11 ;
      always @( DY, DY11 ) TY <= DY[10:0] + DY11 ;
      always @( TX, TY, DRND ) TR = TX * TY + { DRND, 11'b0};
```

```
      assign P11_ = !E_ ? P23 : ( xEND ? TR[11] : R[11]);
      assign P11 = EPL ? 1'bz : P11_;
endmodule
```

5.1.3.15 556PT7A – программируемое постоянное запоминающее устройство.

Микросхема 556PT7A представляет собой программируемое постоянное запоминающее устройство с организацией 2048 8-разрядных слов. В исходном состоянии в незапрограммированных схемах в ячейках память записан логический «0». Запись логической единицы производится путем пережигания нихромовых перемычек. В программной реализации элемента считывание данных из .hex файла, содержащего прошивку микросхемы, ведется при наличии уровня низкого напряжения на входе CS1 и наличии уровней высокого напряжения на входах CS2, CS3 [11].

Программная реализация микросхемы 556PT7A:

```
module ic_556rt7a (CS1, CS2, CS3,A0, A1, A2, A3, A4, A5, A6, A7, A8,
A9, A10,Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7,U);
parameter ADDR_SIZE = 11;
parameter DATA_SIZE = 8;
parameter FILENAME_MEMDATA = "";

input wire CS1, CS2, CS3;
input wire A0, A1, A2, A3, A4, A5, A6, A7, A8, A9, A10;
inout tri Q0, Q1, Q2, Q3, Q4, Q5, Q6, Q7;
input U;

wire CS;
wire [ (ADDR_SIZE - 1) : 0 ] adr;
wire [ (DATA_SIZE - 1) : 0 ] Q;
reg [ (DATA_SIZE - 1) : 0 ] mem [ 0 : ((2**ADDR_SIZE)-1) ];

assign { Q7, Q6, Q5, Q4, Q3, Q2, Q1, Q0 } = Q;
assign adr = { A10, A9, A8, A7, A6, A5, A4, A3, A2, A1, A0 };
assign CS = !CS1 & CS2 & CS3;
assign Q = !CS ? {DATA_SIZE{1'bz}} : mem[adr]; //Inicializacija mas-
sivom dannih

initial
begin
      if (FILENAME_MEMDATA != "")
      begin
            `ifdef QUARTUS  // define in Verilog HDL macro in Assig-
ments->Settings->Verilog HDL input( Assigments -> Settings -> Analysis
and Syntesis Settings -> Verilog HDL input)
                  if (FILENAME_MEMDATA == "mem.data") //trick to
Quartus, not need in iverilog
```

```
                    $readmemh("mem.data", mem);
            `else
                $readmemh(FILENAME_MEMDATA, mem);
            `endif
      end
end
endmodule
```

5.1.3.16 1533КП7 – селектор-мультиплексор на 8 каналов со стробированием.

Микросхема 1533КП7 представляет собой селектор-мультиплексор из 8 в 1 и в зависимости от установленного на входах выбора данных sed1-sed3 кода разрешает прохождение данных на выходы y0 и y0_n только от одного из восьми информационных входов d0-d7, при этом на входе стробирования c0_n должно быть установлено напряжение низкого уровня, при высоком уровне напряжения на нем выход y0 устанавливается в состояние низкого уровня напряжения, а инверсный выход y0_n – в состояние высокого уровня напряжения [8].

Программная реализация микросхемы 1533КП7:

```
module ic_1533kp7
(
      input c0_n,
      input d0, d1, d2, d3, d4, d5, d6, d7,
      input sed1, sed2, sed3,
      output y0, y0_n

);
assign y0_n = ~y0;

wire [7:0] D1 = {d7, d6, d5, d4, d3, d2, d1, d0};
assign y0 = (c0_n == 1'b0)? D1[{sed3, sed2, sed1}] : 1'b0;

endmodule
```

5.1.3.17 533ИП5 – 9-разрядная схема контроля четности/нечетности.

Микросхема 533ИП5 представляет собой девятиразрядную схему контроля четности (нечетности) и позволяет производить сравнительно простой контроль операций, осуществляемых в системе при обработке цифровой информации. В схеме ИЗД.032.677 элементы 533ИП5 (D45, D46, D64) выполняют функцию обнаружения неисправностей.

Программная реализация микросхемы 533ИП5:

```
module ic_533ip5(d0, d1, d2, d3, d4, d5, d6, d7, d8, q, nq);
      input d0, d1, d2, d3, d4, d5, d6, d7, d8;
      output q, nq;

      assign q = ((d0 + d1 + d2 + d3 + d4 + d5 + d6 + d7 + d8) % 2) ?
1'b0 : 1'b1;
      assign nq = ~q;
endmodule
```

5.2 Разработка тестовой программы

5.2.1 Привязка логических интерфейсов к объекту контроля

На основе Netlist’a, сгенерированного при помощи Altera Quartus II, в системе автоматизированного проектирования «SimTest» создается проект, в базу данных загружаются HDL-модели компонентов объекта контроля, разработчик производит распределение контактов краевых разъемов объекта контроля на входные и выходные.

Перед подачей данных на входы сформированной при помощи Altera Quartus II модели диагностируемого изделия осуществляют группировку всех входов объекта контроля на подмножества по признаку - каждому подмножеству подключен один типовой функциональный узел, и каждому такому узлу ставится в соответствие один из имеющийся в базе данных программных модулей - логических интерфейсов. Привязка логических интерфейсов ко входам объекта контроля приведена на рис.11.

Для упрощения генерации теста и сокращения времени на его разработку был применен логический интерфейс «Clock» для требующего постоянных на протяжении всего теста изменений входа X_OP_3 и универсальный логический интерфейс «Группа сигналов» для остальных входов.

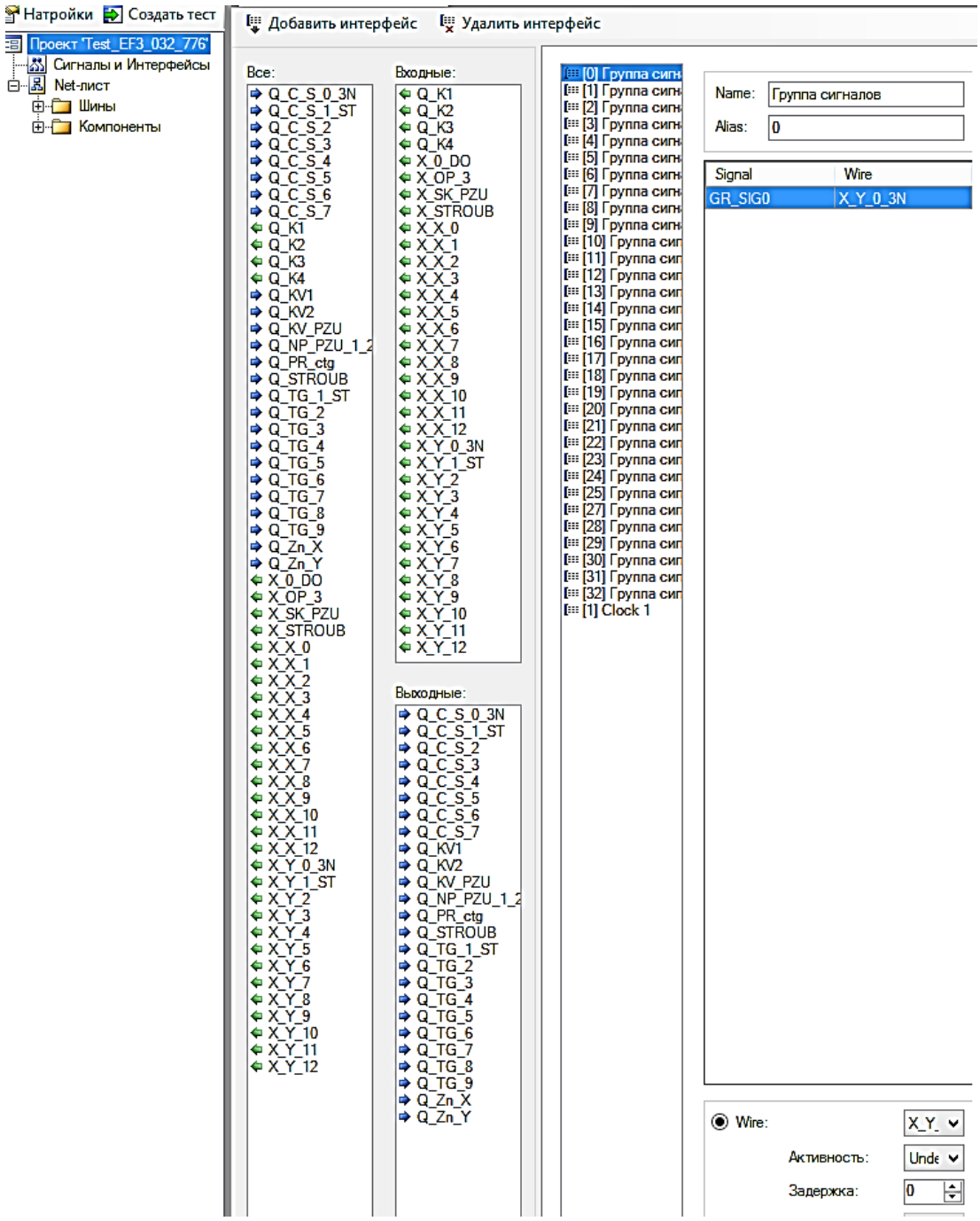

Рис.11 Привязка логических интерфейсов ко входам объекта контроля ИЗД.032.677.

5.2.2 Генерация теста

Формирование входных воздействий осуществляется при помощи логических интерфейсов, которые выполняют команды в соответствии с некоторой программой, написанной на достаточно простом языке управления логическими интерфейсами, которая называется скриптом и представляет собой текстовый файл, содержащий смешанную последовательность команд для логических интерфейсов. Логический интерфейс дешифрирует относящуюся к нему команду и транслирует ее в некоторый фрагмент кода на языке Verilog HDL, представляющий модель выходных сигналов – откликов модели. Скрипт формирования входных воздействий для объекта контроля ИЗД.032.677 приведен в Приложении 2.

Применение логических интерфейсов и скрипта, приведенного в Приложении 2, при формировании теста позволило оперировать высокоуровневыми понятиями «операция логического интерфейса» вместо низкоуровневого двоичного (шестнадцатеричного) представления сочетаний входных сигналов, используемых ранее, что позволило автоматизировать процесс формирования контрольно-диагностических тестов, сократить временные затраты (при разработке теста для объекта контроля ИЗД.032.677 временные затраты были сокращены ~в 3 раза).

5.2.3 Моделирование тест-программы

Моделирование контрольно-диагностического теста осуществляется системой автоматизированного проектирования (САПР) «SimTest» с использованием всех HDL-моделей, относящихся к объекту контроля. Перед началом моделирования все необходимые HDL-модели компонентов, описанные ранее, а также Netlist, сгенерированный средой проектирования Altera Quartus II, загружаются в базу данных САПР «SimTest». По завершению моделирования формируется стандартный .vcd-файл результатов, в который записываются временные последовательности состояний всех сигнальных линий устройства – как на краевых разъемах (входы и выходы объекта контроля), так и в промежуточных точках (рис.12).

Рис.12 Временные последовательности сигнальных линий объекта контроля ИЗД.032.677.

5.3 Предварительная оценка эффективности теста

Система «SimTest» анализирует .vcd-файл результатов моделирования и вычисляет тестовое покрытие для всех уровней схемы объекта контроля (рис.13). Под тестовым покрытием в рассматриваемой системе подразумевается отношение (в процентах) активированных в ходе выполнения теста сигнальных линий диагностируемого изделия $N_{изм}$ к общему числу сигнальных линий N, содержащихся в нем[16] .

$$K = {}^{N_{изм}}/_{N} \cdot 100\% \qquad (25)$$

На рис. 15 показано окно результатов покрытия: слева перечень элементов и каналов объекта контроля ИЗД.032.677, справа статистика по переключению активных элементов и цепей. При этом активированной считается сигнальная линия, которая хотя бы один раз изменила свое состояние с логического «0» на логическую «1», или наоборот [19].

Таким образом, критерием покрытия в системе является покрытие по переключению. Этот критерий определяет качество теста. При разработке тестовых последовательностей необходимо, чтобы покрытие было как можно большим (в идеальном случае 100%).

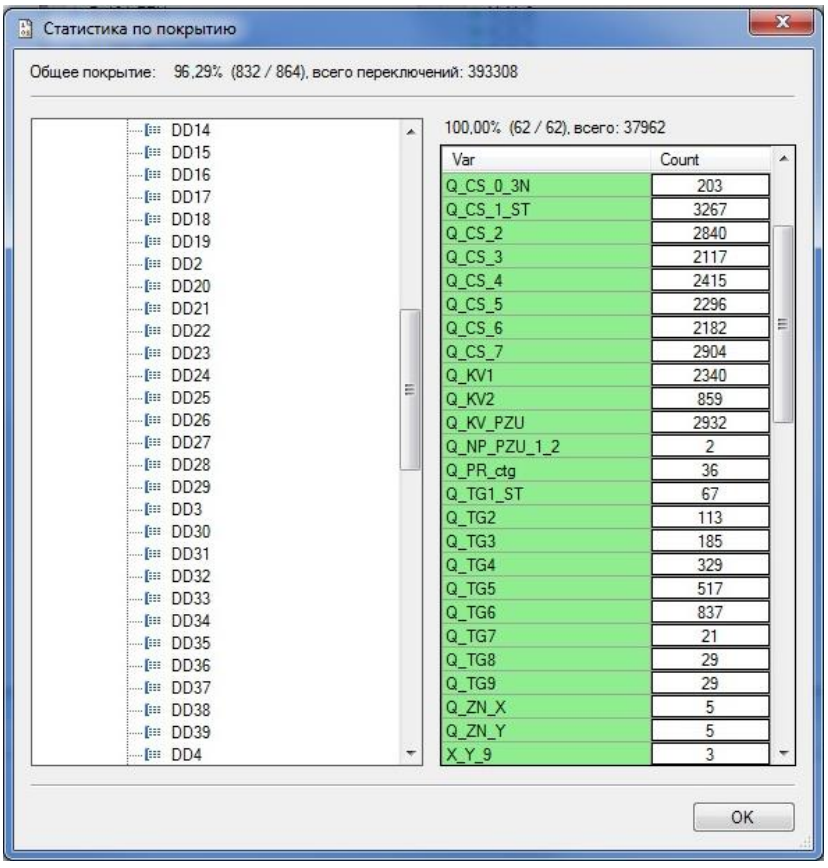

Рис.13 Статистика по покрытию тест-программы для объекта контроля ИЗД.032.677.

Покрытие разработанного теста составляет 96,29%. Оно удовлетворяет заявленным требованиям и является максимальным, т.к. 3,71% составляют

сигнальные линии, которые в силу особенностей схемы не имеют возможности изменения состояния, а также сигнальные линии, подключенные к питанию (постоянная лог. «1») и корпусу (постоянный лог. «0»).

5.4 Отладка тестовой программы на реальном объекте контроля

В случае достижения необходимого уровня покрытия система формирует тест устройства как в текстовом, так и в бинарном виде. Для отладки тестовой программы на реальном объекте контроля в целях упрощения задачи тест загружается в программный пакет «Ястек» (рис.14), взаимодействующий с установкой тестового контроля. На этапе анализа объекта контроля была установлена частота тестирования – 1МГц.

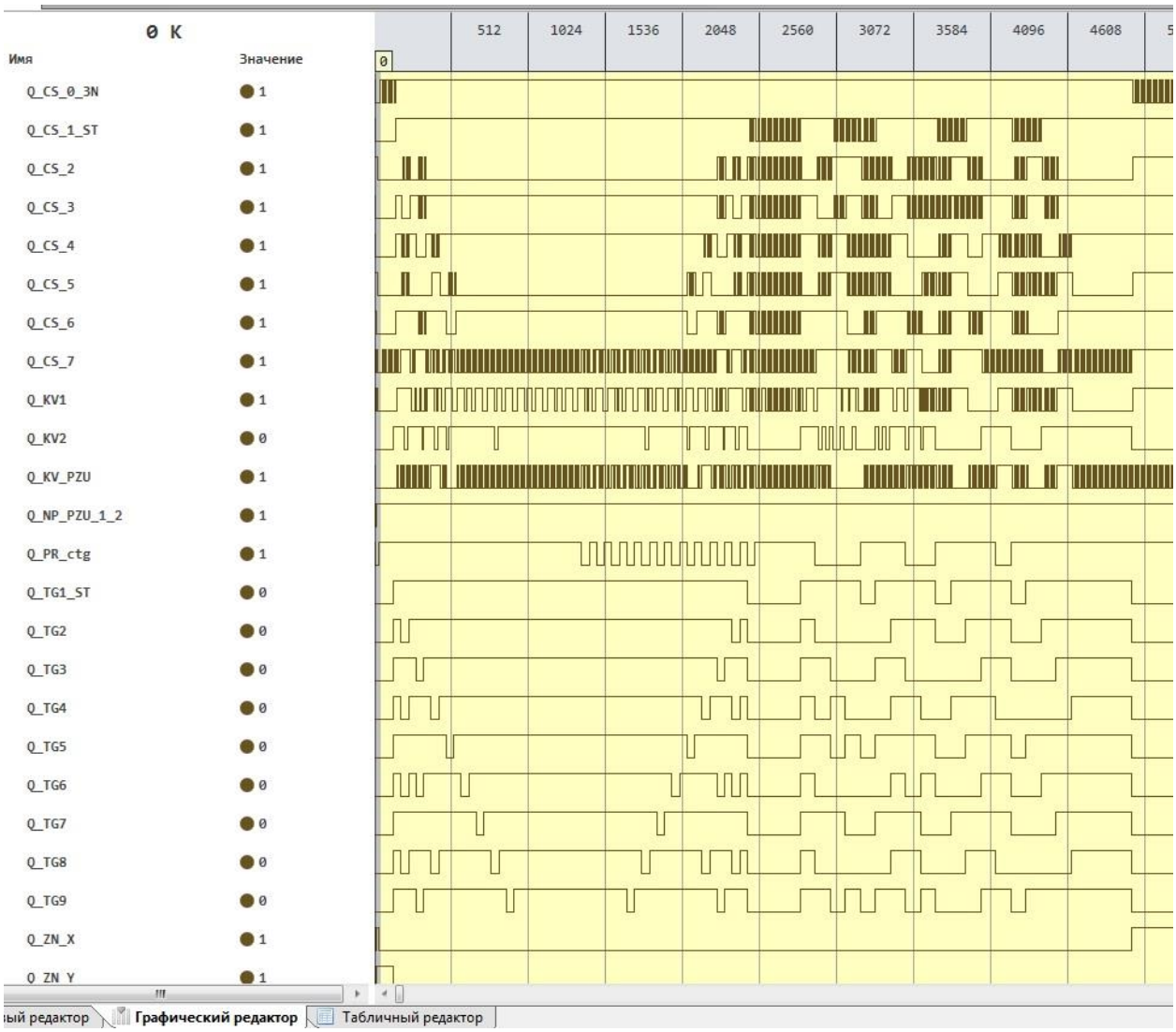

Рис. 14 Фрагмент загруженного в программный пакет «Ястек» теста для объекта контроля ИЗД.032.677.

Отладка тестовой программы осуществляется с помощью установки тестового контроля на эталонном образце изделия (изделие заведомо исправно). Следует отметить, что при отладке теста на установке используется

не только сам полученный тест, но и визуализированные результаты моделирования, позволяющие видеть определенные заранее временные диаграммы сигналов в программном пакете «Ястек». В результате прогона теста в диалоговом окне отображается информация о прохождении теста (рис. 15), на основе которого делается вывод об исправности объекта контроля.

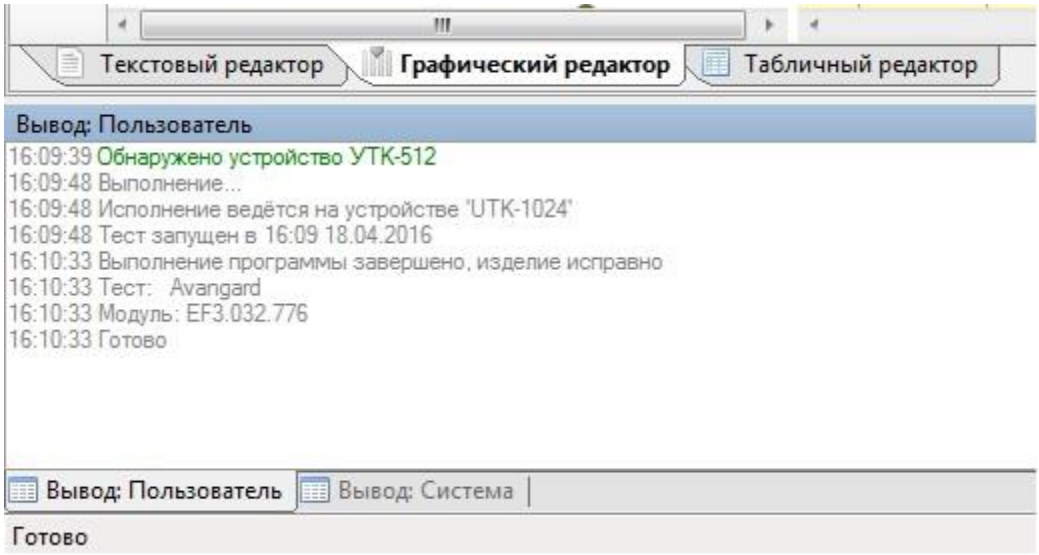

Рис. 15 Результат прогона тестовой программы на установке тестового контроля.

В случае некорректного прохождения теста (останов по браку), в графическом редакторе программного пакета «Ястек» разработчиком определяется, на каком контакте возникла ошибка и принимается решение о дальнейшей доработке теста – корректировке HDL-моделей компонентов или связей между ними, отраженными в Netlist’e.

В случае корректного прохождения тестовая программа принимается в эксплуатацию.

5.5 Анализ трудозатрат

На примере объекта контроля ИЗД.032.677, рассмотренного в главе 5, можно убедиться, что процесс формирования контрольно-диагностического теста – трудоемкая, кропотливая, требующая больших усилий и временных затрат работа. Существенную ее часть составляет этап формирования HDL-моделей компонентов сложных объекта контроля.

В процессе формирования контрольно-диагностических тестов для более, чем 10 объектов контроля пополнялась база HDL-моделей типовых компонентов (содержащихся более, чем в одном изделии). В табл. 4 и на рис. 16, 17 отражена статистика по оценке временных затрат на эмуляцию радиоэлектронных устройств (включающую в себя как формирование модели функционирования объекта контроля в целом, так и ее составных частей – компонентов) и формирование контрольно-диагностических тестов (включающее в себя все этапы, рассмотренные в главе 4). При этом для наглядности временные затраты исчисляются в «д» - рабочих днях.

Таблица 4 – Статистика временных затрат при формировании тестовых программ

Объект контроля	Количество различных компонентов объекта контроля	Временные затраты на формирование HDL-моделей до пополнения базы данных, д	Временные затраты на эмуляцию объекта контроля до пополнения базы данных, д	Временные затраты на эмуляцию объекта контроля после пополнения базы данных, д	Временные затраты на формирование теста до пополнения базы данных, д	Временные затраты на формирование теста после пополнения базы данных, д
Формирователь цифровых сигналов	22	10	12	2	15	5
Схема контроля входных сигналов	13	2	3	1	4	3
Аналого-цифровой преобразователь	29	9	11	3	15	6
Шинный формирователь данных	15	2	3	2	5	3
Дешифратор адреса ОЗУ	17	5	6	2	8	4
Буфер адреса ОЗУ	6	1	1	1	2	1
Буфер данных	24	13	15	3	19	6
Формирователь команд	28	11	14	3	17	7
Схема контроля цифровых сигналов	26	11	13	3	17	6
Формирователь начальной длительности бланка	19	5	7	2	9	4
Общие трудозатраты, д			85	22	111	45

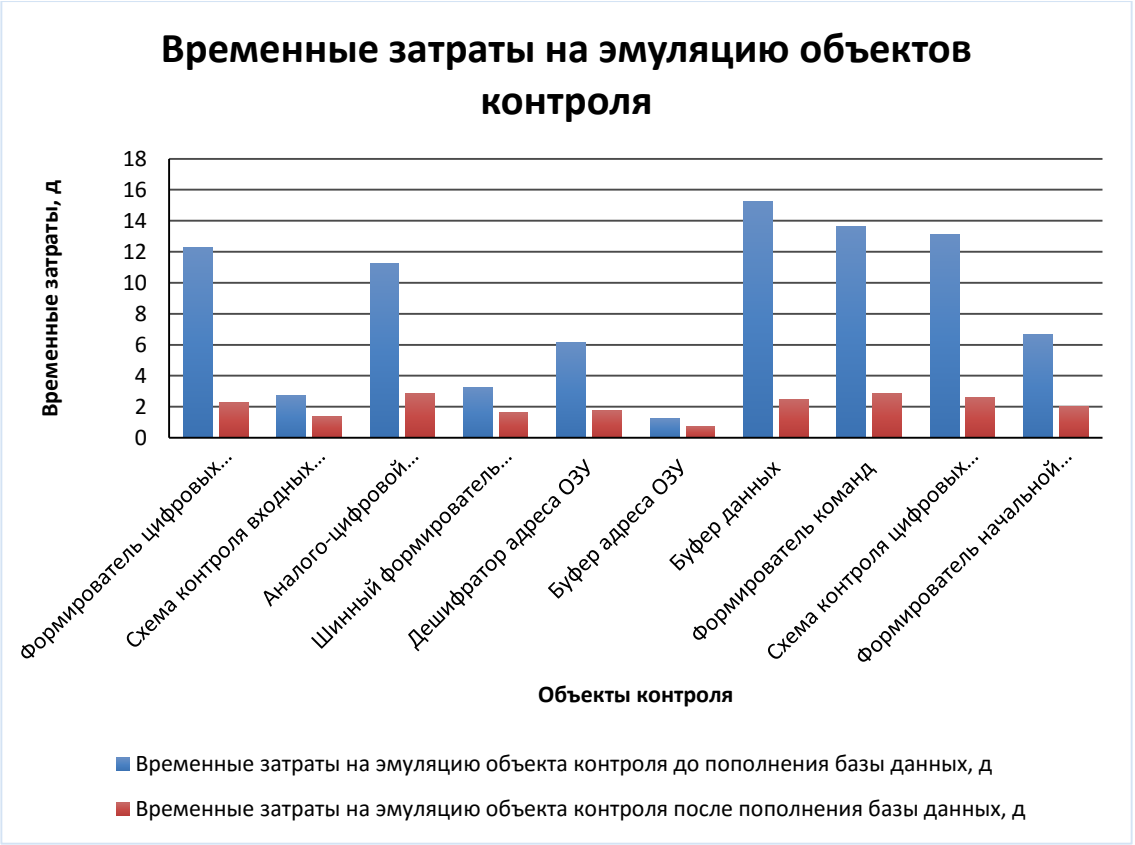

Рис. 16 Временные затраты на эмуляцию объектов контроля

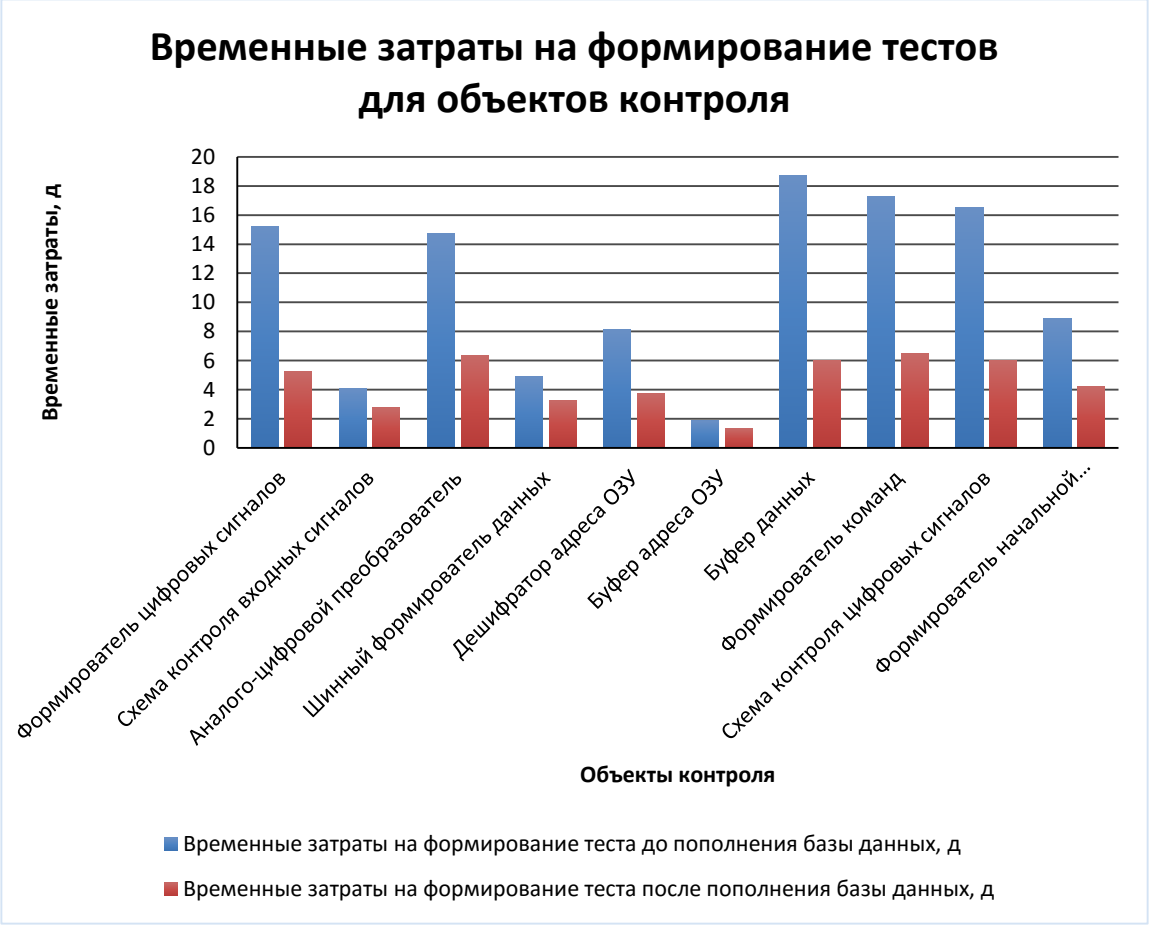

Рис. 17 Временные затраты на формирование тестов для объектов контроля.

Из данных таблицы и диаграмм видно, что пополнение базы HDL-моделей компонентов позволило значительно сокращать время эмуляции цифровых радиоэлектронных устройств – в 3,9 раза и как следствие трудозатраты на формирование тест-программ – в 2,5 раза.

6 Выводы

На примере цифрового объекта контроля ИЗД.032.677 была произведена эмуляция радиоэлектронного устройства: была проанализирована схема устройства, при помощи мультиплатформенной среды проектирования Altera Quartus II была оцифрована схема и сформирована модель функционирования объекта, содержащая информацию о связях компонентов изделия; были разработаны программные модели компонентов объекта контроля на языке Verilog HDL.

При разработке контрольно-диагностического теста была осуществлена привязка имеющихся в базе автоматизированной системы проектирования «SimTest» логических интерфейсов ко входам на краевых разъемах объекта контроля; были сформированы при помощи скрипта входные воздействия; при моделировании тестовой программы были получены отклики модели на выходах изделия.

В результате моделирования теста была произведена оценка эффективности разработанной тестовой программы для цифровой ячейки ИЗД.032.677, получены удовлетворяющие заявленным требованиям показатели тестового покрытия.

При отладке на эталонном (заведомо исправном) объекте контроля была получена информация об исправности изделия, что подтверждает работоспособность разработанного контрольно-диагностического теста.

Была приведена оценка трудоемкости формирования тестовых программ для 10 объектов контроля. Анализ показал, что пополнение базы HDL-моделей компонентов позволило значительно сокращать время эмуляции цифровых радиоэлектронных устройств – в 3,9 раза и как следствие трудозатраты на формирование тест-программ – в 2,5 раза.

7 Заключение

В магистерской диссертации:

- был проанализирован теоретический материал и исследована методика формирования контрольно-диагностических тестов;
- рассмотрены и реализованы этапы формирования контрольно-диагностических тестов с использованием среды проектирования Altera Quartus II, САПР «SimTest», программного пакета «Ястек»;
- приведена оценка эффективности тестовой программы, удовлетворяющая заявленным требованиям;
- приведена статистика по результатам формирования контрольно-диагностических тестов для 10 объектов контроля, а также произведен анализ сокращения временных затрат на эмуляцию радиоэлектронных устройств и формирование тестовых программ при пополнении базы HDL-компонентов.

8 Список литературы

[1] Отраслевой стандарт ОСТ4 ГО.303.000, 1972. Узлы и блоки радиоэлектронной аппаратуры. Методы составления контрольных тестов цифровых узлов.

[2] Отраслевой стандарт ОСТ4 ГО.303.001, 1973. Узлы и блоки радиоэлектронной аппаратуры. Методика диагностики цифровых узлов.

[3] Государственный стандарт ГОСТ Р 55692-2013: МОДУЛИ ЭЛЕКТРОННЫЕ. Методы составления и отладки тест-программ

[4] Государственный стандарт ГОСТ 2091-89: Техническая диагностика термины и определения.

[5] Патент RU 2475821 C1 от 20.02.2013г., МПК G06F11/22, G05B23/00 Способ предварительной оценки качества диагностических тестов

[6] Надежность и эффективность в технике. Справочник в 10 т. т.9. Техническая диагностика. Под ред. В.В.Клюева, М.: Машиностроение, 1987, с.177-178.

[7] Поляков А.К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры. – М., СОЛОН-Пресс, 2003.

[8] Шило В.Л. Популярные цифровые микросхемы. Справочник. М.: “Радио и связь”, 1987.

[9] Петровский И.И., Прибыльский А.В., Троян А.А. и др. Логические ИС КР1533, КР1554 часть 1, часть 2. Справочник. М.: Бином, 1993.

[10] Опадчий Ю.Ф. Общая технология проектирования в среде Quartus II. Методические указания по курсу "Схемотехническое проектирование ЭВС".

[11] Michael L. Bushnell, Vishwani D. Agrawal. Essentials of electronic testing for digital, memory and mixed-signal VLSI circuits. New York, Boston, Dordrecht, London, Moscow. Kluwer Academic Publishers, 2000.

[12] Елаев Е.В. / Интерфейсный метод автоматизированной генерации тестовых воздействий для цифровых радиоэлектронных объектов контроля // Вестник Санкт-Петербургского государственного университета технологии и дизайна. Серия 1: Естественные и технические науки. 2015. № 4. С. 19-24.

[13] Мащинский Н.С., Елаев Е.В., Федюкович П.А. / Моделирование сложных цифровых устройств с целью их тестирования // Процессы управления и устойчивость. 2015. Т. 2. № 1. С. 452-457.

[14] Grishkin V., Yelaev Y., Lopatkin G., Mikhailov A., Ovsyannikov D. / Interface method of digital devices testing // Tenth International Vacuum Electron Sources Conference (IVESC) & Second International Conference on Emission Electronics (ICEE) 2014. С. 107-108.

[15] Melnik V.I., Mikhailov A.N., Grishkin V.M., Ovsyannikov D.A., Yelaev Y.V. / Methods of modeling of the test inputs for analysis the digital devices // 2014 International conference on computer technologies in physical and engineering applications (ICCTPEA) Editor: E. I. Veremey. Санкт-Петербургский государственный университет; IEEE (IEEE Catalog number CFP14BDA-USB). 2014. С. 112-113.

[16] Мельник В., Гришкин В., Михайлов А., Овсянников Д. / Методика разработки тест-программ контроля и диагностики цифровых устройств с использованием САПР SimTest // Электроника: Наука, технология, бизнес. 2013. № S (128). С. 118-124.

[17] Михайлов А., Мельник В., Овсянников Д. / Тестовый контроль и диагностика радиоэлектронной аппаратуры // Электроника: Наука, технология, бизнес. 2013. № S (128). С. 114-117.

[18] Степанов Ю.Л., Гришкин В.М., Лопаткин Г.С., Большаков А.А., Ким М.А. / Автоматизированное построение тестов цифровых электронных модулей для комплекса тестового контроля и диагностики УТК-512. // Вопросы радиоэлектроники, сер. Общетехническая, 2012, вып.1, с.79-89.

[19] Гришкин В.М., Степанов Ю.Л., Лопаткин Г.С., Большаков А.А. / Подход к разработке тестов цифровых электронных модулей для автоматического тестового оборудования //Вопросы радиоэлектроники. 2013. Т. 1. № 1. С. 89-99.

Приложение 1.

533ИМ6.

```
     module ic_533IM6 ( A0, A1, A2, A3, B0, B1, B2, B3, C0, S0, S1,
S2, S3, C4);
     input C0, A0, A1, A2, A3, B0, B1, B2, B3;
     output S0, S1, S2, S3, C4;

     ic_74S283 add (C0, {A3,A2,A1,A0}, {B3,B2,B1,B0}, {S3,S2,S1,S0},
C4);

     endmodule

     module ic_74S283 (C0, A, B, S, C4);

     input[3:0]    A, B;
     input         C0;

     output[3:0]   S;
     output        C4;

     TopLevel74283 Ckt74283 (C0, A, B, S, C4);

     endmodule

     module TopLevel74283 (C0, A, B, S, C4);

     input[3:0]    A, B;
     input         C0;
     output[3:0]   S;
     output        C4;
     wire[3:0]     GB, PB, AxB;
     wire[3:0]     C;

     GP_Module GP_Mod1(A, B, GB, PB, AxB);
     CLA_Module CLA_Mod2(GB, PB, C0, C, C4);
     Sum_Module Sum_Mod3(AxB, C, S);

     endmodule

     module GP_Module(A, B, GB, PB, AxB);
     input[3:0]    A, B;
     output[3:0]   GB, PB, AxB;
     wire[3:0]     P;

     nor PBgate0(PB[0], A[0], B[0]);
     nand GBgate0(GB[0], A[0], B[0]);
     not Pgate0(P[0], PB[0]);
     and AxBgate0(AxB[0], GB[0], P[0]);

     nor PBgate1(PB[1], A[1], B[1]);
     nand GBgate1(GB[1], A[1], B[1]);
     not Pgate1(P[1], PB[1]);
     and AxBgate1(AxB[1], GB[1], P[1]);

     nor PBgate2(PB[2], A[2], B[2]);
     nand GBgate2(GB[2], A[2], B[2]);
```

```
not Pgate2(P[2], PB[2]);
and AxBgate2(AxB[2], GB[2], P[2]);

nor PBgate3(PB[3], A[3], B[3]);
nand GBgate3(GB[3], A[3], B[3]);
not Pgate3(P[3], PB[3]);
and AxBgate3(AxB[3], GB[3], P[3]);

endmodule

module   CLA_Module(GB, PB, C0, C, C4);
input[3:0]      GB, PB;
input           C0;
output[3:0]     C;
output          C4;

not C0Bgate(C0B, C0);
not C0gate(C[0], C0B);

buf PB0gate(PB0, PB[0]);
and C0BGB0gate(C0BGB0, C0B, GB[0]);

buf PB1gate(PB1, PB[1]);
and PB0GB1gate(PB0GB1, PB[0], GB[1]);
and C0BGB01gate(C0BGB01, C0B, GB[0], GB[1]);

buf PB2gate(PB2, PB[2]);
and PB1GB2gate(PB1GB2, PB[1], GB[2]);
and PB0GB12gate(PB0GB12, PB[0], GB[1], GB[2]);
and C0BGB012gate(C0BGB012, C0B, GB[0], GB[1], GB[2]);

buf PB3gate(PB3, PB[3]);
and PB2GB3gate(PB2GB3, PB[2], GB[3]);
and PB1GB23gate(PB1GB23, PB[1], GB[2], GB[3]);
and PB0GB123gate(PB0GB123, PB[0], GB[1], GB[2], GB[3]);
and C0BGB0123gate(C0BGB0123, C0B, GB[0], GB[1], GB[2], GB[3]);

nor C4gate(C4,PB3,PB2GB3,PB1GB23,PB0GB123,C0BGB0123);

nor C3gate(C[3],PB2,PB1GB2,PB0GB12,C0BGB012);

nor C2gate(C[2],PB1,PB0GB1,C0BGB01);

nor C1gate(C[1],PB0,C0BGB0);


endmodule

module   Sum_Module(AxB, C, S);
input[3:0]      AxB;
input[3:0]      C;
output[3:0]     S;

xor Sum0(S[0], C[0], AxB[0]);
xor Sum1(S[1], C[1], AxB[1]);
xor Sum2(S[2], C[2], AxB[2]);
xor Sum3(S[3], C[3], AxB[3]);
endmodule
```

Приложение 2.

Скрипт формирования входных воздействий для объекта контроля ИЗД.032.677.

```
#GROUP_SIG 0 //X_Y_0_3N
val 0 1
val 100 0
val 6100 1
val 7000 0
//
#GROUP_SIG 1 //X_Y_1_ST
val 0 0
val 50 1
val 2450 0
val 2700 1
//
#GROUP_SIG 2 //X_Y_2
val 0 0
val 150 1
val 2350 0
val 3200 1
//
#GROUP_SIG 3 //X_Y_3
val 0 0
val 250 1
val 2250 0
val 3700 1
//
#GROUP_SIG 4 //X_Y_4
val 0 0
val 350 1
val 2150 0
val 4200 1
//
#GROUP_SIG 5 //X_Y_5
val 0 0
val 450 1
val 2050 0
val 2700 1
//
#GROUP_SIG 6 //X_Y_6
val 0 0
```

```
val 550 1
val 1950 0
val 3200 1
//
#GROUP_SIG 7 //X_Y_7
val 0 0
val 650 1
val 1850 0
val 3700 1
//
#GROUP_SIG 8 //X_Y_8
val 0 0
val 750 1
val 1750 0
val 4200 1
//
#GROUP_SIG 9 //X_Y_9
val 0 0
val 850 1
val 1650 0
val 2700 1
//
#GROUP_SIG 10 //X_Y_10
val 0 0
val 950 1
val 1550 0
val 3200 1
//
#GROUP_SIG 11 //X_Y_11
val 0 0
val 1050 1
val 1450 0
val 3700 1
//
#GROUP_SIG 12 //X_Y_12
val 0 0
val 1150 1
val 1350 0
val 4200 1
//
#GROUP_SIG 13 //X_X_0
val 0 0
```

```
val 5000 1
val 6000 0
//
#GROUP_SIG 14 //X_X_1
val 0 0
val 100 1
val 2500 0
val 2800 1
val 3199 0
val 3300 1
val 3699 0
val 3800 1
val 4199 0
val 4300 1
//
#GROUP_SIG 15 //X_X_2
val 0 0
val 200 1
val 2400 0
val 2900 1
val 3199 0
val 3400 1
val 3699 0
val 3900 1
val 4199 0
val 4400 1
//
#GROUP_SIG 16 //X_X_3
val 0 0
val 300 1
val 2300 0
val 3000 1
val 3199 0
val 3500 1
val 3699 0
val 4000 1
val 4199 0
val 4500 1
//
#GROUP_SIG 17 //X_X_4
val 0 0
val 400 1
```

```
val 2200 0
val 3100 1
val 3199 0
val 3600 1
val 3699 0
val 4100 1
val 4199 0
val 4600 1
//
#GROUP_SIG 18 //X_X_5
val 0 0
val 500 1
val 2100 0
val 2800 1
val 3199 0
val 3300 1
val 3699 0
val 3800 1
val 4199 0
val 4300 1
//
#GROUP_SIG 19 //X_X_6
val 0 0
val 600 1
val 2000 0
val 2900 1
val 3199 0
val 3400 1
val 3699 0
val 3900 1
val 4199 0
val 4400 1
//
#GROUP_SIG 20 //X_X_7
val 0 0
val 700 1
val 1900 0
val 3000 1
val 3199 0
val 3500 1
val 3699 0
val 4000 1
```

```
val 4199 0
val 4500 1
//
#GROUP_SIG 21 //X_X_8
val 0 0
val 800 1
val 1800 0
val 3100 1
val 3199 0
val 3600 1
val 3699 0
val 4100 1
val 4199 0
val 4600 1
//
#GROUP_SIG 22 //X_X_9
val 0 0
val 900 1
val 1700 0
val 2800 1
val 3199 0
val 3300 1
val 3699 0
val 3800 1
val 4199 0
val 4300 1
///
#GROUP_SIG 23 //X_X_10
val 0 0
val 1000 1
val 1600 0
val 2900 1
val 3199 0
val 3400 1
val 3699 0
val 3900 1
val 4199 0
val 4400 1
//
#GROUP_SIG 24 //X_X_11
val 0 0
val 1100 1
```

```
val 1500 0
val 3000 1
val 3199 0
val 3500 1
val 3699 0
val 4000 1
val 4199 0
val 4500 1
//
#GROUP_SIG 25 //X_X_12
val 0 0
val 1200 1
val 1400 0
val 3100 1
val 3199 0
val 3600 1
val 3699 0
val 4100 1
val 4199 0
val 4600 1
//
#GROUP_SIG 27 //X_SK_PZU
val 0 1
val 1 0
val 2 1
val 10000 0
//
#GROUP_SIG 28 //X_0_DO
val 0 1
val 1 0
val 2 1
//
#GROUP_SIG 29 //Q_K1
val 0 1
val 5400 0
val 9000 1
//
#GROUP_SIG 30 //Q_K2
Base 0
Inc 1
Delay 0
Tcount 25
```

```
Tend 3650
//
#GROUP_SIG 31 //Q_K3
Base 0
Inc 1
Delay 0
Tcount 50
Tend 3650
//
#GROUP_SIG 32 //Q_K4
Base 0
Inc 1
Delay 0
Tcount 100
Tend 3600
//
```